Titre: Title:	Modélisation de la conduction neuronale périphérique dédiée à l'amélioration de la stimulation sélective
Auteur: Author:	Dominique Paquet-Ferron
Date:	2006
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Paquet-Ferron, D. (2006). Modélisation de la conduction neuronale périphérique dédiée à l'amélioration de la stimulation sélective [Mémoire de maîtrise, École Polytechnique de Montréal]. PolyPublie. https://publications.polymtl.ca/7660/

URL de PolyPublie: PolyPublie URL:	https://publications.polymtl.ca/7660/
Directeurs de recherche: Advisors:	
Programme: Program:	Non spécifié

UNIVERSITÉ DE MONTRÉAL

MODÉLISATION DE LA CONDUCTION NEURONALE PÉRIPHÉRIQUE DÉDIÉE À L'AMÉLIORATION DE LA STIMULATION SÉLECTIVE

DOMINIQUE PAQUET-FERRON
INSTITUT DE GÉNIE BIOMÉDICAL
ÉCOLE POLYTECHNIQUE DE MONTRÉAL

MÉMOIRE PRÉSENTÉ EN VUE DE L'OBTENTION
DU DIPLÔME DE MAÎTRISE ÈS SCIENCES APPLIQUÉES
(GÉNIE BIOMÉDICAL)
JANVIER 2006

Your file Votre référence
ISBN: 978-0-494-16829-5
Our file Notre référence
ISBN: 978-0-494-16829-5

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Ce mémoire intitulé :

MODÉLISATION DE LA CONDUCTION NEURONALE PÉRIPHÉRIQUE DÉDIÉE À L'AMÉLIORATION DE LA STIMULATION SÉLECTIVE

Présenté par : PAQUET-FERRON, Dominique

en vue de l'obtention du diplôme de : Maîtrise ès sciences appliquées

a été dûment accepté par le jury d'examen constitué de :

M. KHOUAS Abdelhakim, Doct., président

M. SAWAN Mohamad, Ph.D., membre et directeur de recherche

M. MATHIEU PIERRE A., D.Sc.A., membre

REMERCIEMENTS

Je tiens tout d'abord à remercier mon directeur de recherche M. Mohamad Sawan pour m'avoir intégré à son équipe et m'avoir permis de réaliser ce projet. Je tiens à le remercier de m'avoir guidé tout au long de ce périple mais, surtout pour m'avoir permis d'en arriver ici aujourd'hui.

J'aimerais également remercier toute l'équipe Polystim ainsi que celle d'Urostim pour l'aide et le support qu'ils m'ont procuré tout au long de cette aventure; je pense ici à Fayçal et à Jonathan pour leur conseils prodigieux, à Nicolas, Tommy, Benoit, Jean-Francois (bis), Gérard et Amer pour leur support moral et la vie au lab. Je voudrais remercier tout particulièrement les stagiaires Germain, Sleiman et Simon pour leur contribution dans ce projet. Je ne peux évidemment pas oublier le support informatique assuré par Réjean et Alex sans qui je n'aurais pas pu réaliser ce projet. Je tiens également à souligner le travail exceptionnel effectué par les secrétaires de notre groupe Ghyslaine et Claudine.

Je voudrais également remercier ma conjointe Véronique et ma fille Élisabeth de m'avoir épaulé tout au long de ce périple et ce, même si plusieurs sacrifices ont été nécessaires. Je voudrais également remercier mes parents pour m'avoir soutenu et encouragé depuis mon enfance.

Pour terminer, j'aimerais remercier, au nom de toute l'équipe, l'Institut de Recherche en Santé du Canada (IRSC) pour le financement accordé à notre équipe dans le cadre de ce projet.

RÉSUMÉ

L'apparition de micro-stimulateurs implantables dans le corps humain est devenue accessible sur le marché. Ces stimulateurs ont pour but de restaurer des fonctions corporelles diverses par le biais de la stimulation électrique. Nous nous intéresserons dans ce travail à la restauration des fonctions urinaires chez les blessés médullaires. Plusieurs équipes ont déjà montré qu'il était possible d'engendrer une contraction des muscles entourant la vessie (détrusor) à l'aide d'une stimulation électrique neuronale de S2, un nerf émergent de la racine sacrée. Malheureusement, la stimulation à ce niveau engendre une contraction simultanée du sphincter urétral externe et du détrusor, inhibant ainsi la miction. L'équipe Polystim met au point, depuis un peu plus d'une décennie, des stimulateurs implantables dédiés à la récupération des deux principales fonctions urinaires; rétention et évacuation de l'urine.

La stimulation sélective par blocage haute fréquence est une technique utilisée pour réduire la dyssynergie entre le sphincter et le détrusor afin de favoriser la miction. Cette stimulation sélective utilise deux trains de stimuli de fréquences différentes afin d'obtenir une bonne contraction du détrusor tout en inhibant la contraction du sphincter externe. Un modèle informatique du nerf de la racine sacrée à été réalisé dans le cadre de cette maîtrise afin d'optimiser les paramètres de la stimulation sélective. Le modèle est flexible et peut être adapté pour examiner de nouvelles techniques de stimulation. Il est possible d'ajuster les paramètres afin de représenter divers nerfs pour d'autre fonction d'organes.

Le modèle a été réalisé avec les logiciels Neuron et SCIRun/BioPSE; Neuron permet de simuler le comportement des axones et des neurones et BioFEM sert à la résolution d'éléments finis. Ce dernier est utilisé pour calculer la propagation du courant provenant des électrodes au travers des différentes couches entourant le nerf. La comparaison des résultats de simulations avec des enregistrements expérimentaux nous a permis de valider le modèle proposé.

ABSTRACT

Recent work in microelectronics has allowed a generation of implantable stimulators aimed at restoring many vital functions in the human body. Stimulators use the existing nerve pathways in the body to activate and control organs and/or muscles. In this work, we focus on reestablishing bladder functions for people with spinal cord injuries (SCI).

Much research has demonstrated the possible activation of muscles surrounding the bladder (detrusor) by stimulating one of the sacral nerves (S2). Unfortunately, this electrical neurostimulation also activates the contraction of the external sphincter which in turn blocks the flow of urine through the urethra. This phenomenon is called bladder dyssynergia. The Polystim group has been working nearly a decade on an implantable stimulator dedicated to restore both principal bladder functions; retention and evacuation.

Selective stimulation achieved by high frequency blockade is a technique used to reduce bladder dyssynergia. The selective stimulation is composed of two bipolar signals with two distinct frequencies. The lowest frequency is used to obtain a strong bladder contraction and the highest frequency is used to inhibit the sphincter's contraction. This technique, when applied in chronic case experiments, has demonstrated in excess of 50% voiding improvement. A nerve model is proposed in this master's thesis in order to validate and improve the selectivity. The proposed model is easily configurable (for any type of nerves) and allows users to readily modify parameters, if required.

The model has been developed using Neuron and SCIRun/BioPSE. Neuron is used to simulate the axon's and neuron's behavior. BioFEM is used to solve Finite Element Meshes. The purpose of finite element analysis is to compute the current propagation throughout the nerve different layers (epineurium, perineurium, etc). Additionally, simulation results were compared to experimental records in order to validate our model.

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

LISTE DES ANNEXES

INTRODUCTION

L'utilisation de décharges électriques pour le traitement de la douleur a été rapportée au début du 17^{e} siècle, mais c'est en 1791 qu'une première étude fut publiée par Luigi Galvani sur le phénomène de la stimulation électrique. Galvani avait accidentellement trouvé que la contraction d'un muscle pouvait se produire en touchant le nerf relié à celui-ci; il s'est par la suite assuré que ce phénomène était bien provoqué par l'électricité.

Depuis quelques décennies, plusieurs équipes de recherche examinent les fondements de la stimulation électrique neuronale et travaillent aussi à améliorer les techniques de stimulation existantes. On cherche à trouver la meilleure façon de remplacer les impulsions que le cerveau envoit normalement aux différents organes et membres. La stimulation électrique neuronale est une solution d'avenir pour les blessés médullaires; un exemple connu est celui de redonner le contrôle de la vessie à ceux qui ont subi un accident menant à une paralysie des membres inférieurs (paraplégiques). Cette paralysie implique une section de la moelle épinière au niveau lombaire coupant la liaison entre le cerveau et les différents organes/membres situés dans la portion inférieure du corps humain. Notons, parmi plusieurs problèmes, l'incontinence urinaire, la perte des fonctionnalités érectiles, etc.

Les chercheurs ont mis au point toute une panoplie de stimulateurs permettant de redonner une qualité de vie jamais égalée à plusieurs personnes; pensons aux stimulateurs cardiaques pour régulariser le rythme du cœur, aux implants cochléaires et à bien d'autres stimulateurs pour réduire la douleur chronique, assister la respiration, contrôler la fermeture de la main, etc. Cependant, un bon nombre de dysfonctions, tel la perte des fonctions urinaires (évacuation et rétention) ne peuvent être restitués autrement que par chirurgie ou par médication. La perte du contrôle volontaire de la vessie survient généralement suite à un accident avec lésion de la moelle au niveau D12 ou supérieur.

La stimulation électrique des racines sacrées (S1 à S4) montre des résultats très promoteurs car elles contiennent des fibres nerveuses qui innervent à la fois le muscle vésical (détrusor) et le sphincter urétral externe. Une excitation de ces nerfs provoque une contraction simultanée du détrusor et du sphincter; cette simultanéité engendre un problème connu sous le nom de dyssynergie; il s'agit d'un manque de coordination entre la contraction de la vessie et le relâchement du sphincter afin de permettre une bonne miction. Afin de résoudre ce problème, plusieurs solutions ont été développées; celle utilisé par notre équipe (Polystim) est une stimulation sélective grâce à un blocage haute fréquence. Plus précisément, elle consiste à exciter le nerf avec deux trains d'impulsions de courant bipolaires superposés; un train dit « basse fréquence » avec « haute amplitude » permettant de stimuler la vessie et un autre « haute fréquence » avec « basse amplitude » permettant le relâchement du sphincter. Le modèle du nerf présenté dans le présent ouvrage permet de valider la technique d'un point de vue plus théorique.

Cette validation théorique devient de plus en plus nécessaire puisque les tests effectués avec des animaux sont très coûteux et très complexe. En effet, afin de réaliser ces expérimentations, il est primordial d'obtenir les approbations nécessaires auprès des instituts et organismes concernés. Les demandes en ce sens sont soumises à des comités d'éthiques qui doivent évaluer la pertinence des tests, la faisabilité du projet dans son ensemble, mais tout en gardant à l'esprit que dans le cas d'une réussite, le projet serait bénéfique pour beaucoup d'êtres humains. Bref, le processus afin de réaliser des tests in-vivo est très complexe. Cette complexité augmente exponentiellement lorsqu'on désire poursuivre à des stades plus élevés. De plus, les ordinateurs d'aujourd'hui sont de plus en plus performants et les logiciels développés le sont également. On peut obtenir aujourd'hui grâce à la puissance de calcul des ordinateurs des simulations qui dépassent ce que nous croyions qu'il étais un jours possible de réaliser. Pensons par exemple au déchiffrement du génome humains; chose impensable et irréalisable avant la venue d'ordinateurs suffisamment puissants. Pensons également à des projets de plus en plus fréquents où la puissance de calcul nécessaire est tellement élevée qu'on subdivise le

problème en milliers de sous problèmes qu'on éparpille à travers plusieurs ordinateurs à travers le monde. La puissance de calcul disponible aujourd'hui est incomparable à celle d'il y a 10 ans et probablement incomparable à celle qui sera disponible dans 10 ans.

Tel que mentionné précédemment, cette puissance de calcul a permis l'apparition de plusieurs logiciels qui peuvent utiliser la majeure partie des ressources afin de les appliquer à la résolution de problèmes concrets. La résolution des éléments finis est devenue aujourd'hui pratique courante, mais surtout une technique rapide et efficace. On peut modéliser et simuler en quelques minutes ce qui autrefois prenait des semaines à réaliser. Cette prouesse technologique a également permis aux scientifiques de s'intéresser à des problèmes beaucoup plus complexes et plus globaux. Le corps humain, par exemple, présente beaucoup de défis de par sa complexité et, on tente de plus en plus d'en percer les mystères. Le domaine de la neurostimulation est en pleine effervescence et permettra d'obtenir plusieurs avancés technologiques importantes; il est donc primordial d'avoir des modèles robustes et fiables sur lequel nous pouvons développer plusieurs idées à moindre coût qu'avec des tests in-vivo.

Le présent mémoire traite, dans un premier chapitre, du système urinaire, des effets d'un traumatisme spinal sur les différents organes, de notions approfondies sur la constitution physiologique des nerfs ainsi que de la stimulation électrique afin d'obtenir une bonne miction. Le second chapitre aborde les travaux qui ont déjà été réalisés, les travaux présentement en cours ainsi qu'une revue de littérature. Le troisième chapitre présente les différents logiciels utilisés ainsi que leurs rôle respectif. La réalisation et la vérification du modèle sont traitées au quatrième chapitre. Les résultats et la validation sont détaillés dans le cinquième chapitre. Viennent ensuite recommandations, conclusion et annexes.

Chapitre 1

LE SYSTÈME URINAIRE ET LA STIMULATION NEURONALE

Afin de concevoir un modèle réaliste, nous avons entrepris une étude détaillée de la physiologie des nerfs et plus particulièrement, ceux innervant le système urinaire. Il est par conséquent important d'approfondir certaines notions d'anatomie de base du système urinaire ainsi que sur l'organisation du système nerveux. En plus de faire un survol de ces notions, ce chapitre trait également des techniques de stimulations existantes.

1.1 Le système urinaire

Le système urinaire regroupe les organes qui permettent de filtrer, récupérer, emmagasiner et évacuer les déchets produits par toutes les parties du corps. La première étape est effectuée par les reins; il s'agit de filtrer le sang et ainsi le débarrasser de ses déchets et toxines. Ensuite, ils acheminent ces déchets (ainsi que le surplus d'eau) vers la vessie par les canaux collecteurs d'urine (bassinets) qui se prolongent en deux fins canaux (uretères) vers la vessie (Figure 1-1).

1.1.1 Portion inférieure du système urinaire

Les uretères rejoignent la vessie en traversant obliquement la paroi musculaire (hiatus urétral). L'urine est évacuée par l'urètre (situé à l'opposé des deux uretères) vers le méat urinaire (orifice externe de l'urètre). Chez l'homme, l'urètre est généralement plus long que chez la femme et son parcours passe au travers de la prostate et du pénis. Les deux uretères et l'urètre passant autour de la vessie forment une entité topographique habituellement dénommée « trigone ». Il est important de mentionner que cette portion de

la vessie change de morphologie pendant l'évacuation de l'urine; il en sera question lors de la description de la miction.

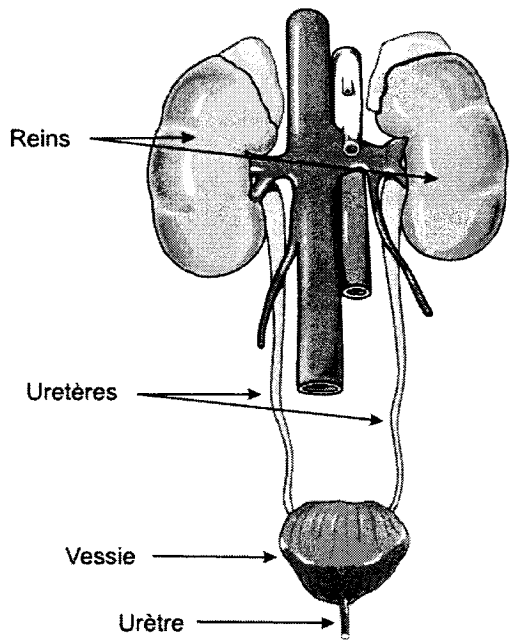


Figure 1-1 : Système urinaire simplifié

1.1.2 Les différents tissues musculaires

Il existe principalement 3 muscles qui servent à coordonner la rétention et l'évacuation de l'urine; le détrusor, le sphincter lisse et le sphincter strié. La vessie est composée de trois couches musculaires, appelé détrusor, essentiellement constitués de fibres musculaires lisses qui permettent de réaliser la fonction contractile de la vessie. Le sphincter lisse est tout simplement un prolongement du détrusor autour du col de la vessie et de l'urètre; il est à noter que les muscles dit « lisses » sont généralement plus enclin à maintenir une contraction longue et moins sujets à la fatigue. Quant au sphincter strié, il entoure le sphincter interne et descend légèrement (environ 2 à 3 cm) au dessous du col. C'est le sphincter strié qui est responsable de la continence active (i.e. consciente) lorsque la vessie atteint sa pleine capacité et que le besoin d'uriner se fait sentir. Il est important de souligner qu'il existe d'autres muscles qui régissent la miction, mais qu'ils

sont considérés ici comme jouant un rôle mineur; surtout pour des blessés médullaires qui n'ont aucun contrôle volontaire de ces muscles.

1.2 Effets d'un traumatisme spinal

Une lésion de la mœlle épinière brise la connexion entre le système nerveux central (SNC) et le système nerveux périphérique (SNP) ce qui entraîne que plusieurs organes/membres cessent de fonctionner correctement. Par exemple, la perte de contrôle volontaire du sphincter strié entraîne des problèmes d'incontinence.

En coupant le lien entre le cerveau et le système périphérique, la vessie n'est plus régit par celui-ci et un déséquilibre entre les influx inhibiteur et facilitateur survient. La vessie n'est donc plus que régulée par ses centres réflexes. L'hyperactivité apparaît car les influx facilitateurs induits par les neurones sensitifs de la paroi vésicale ne peuvent être contrebalancés par les influx inhibiteurs supra-médullaire. Cette hyperactivité augmente la pression vésicale au moment du remplissage et par le fait même, réduit la capacité de stockage de la vessie [29].

D'autres effets indirects peuvent également survenir suite à une lésion de la moelle; au niveau de la vessie par exemple, on assiste souvent à un reflux d'urine de la vessie vers les reins (causé principalement par l'absence des flux inhibiteurs). Ce reflux additionné à un problème de rétention chronique conduit habituellement à une défaillance rénale. L'incontinence, habituellement « traité » par cathétérisme intermittent, conjugué à la stase urinaire favorise les infections urinaires répétitives. L'ensemble des problèmes observés conduisent souvent à une défaillance des fonctions rénales; cause importante de décès chez les tétraplégiques [25].

1.3 Les voies neuronales

Afin de réaliser pleinement l'objectif de miction par stimulation électrique, il est nécessaire d'identifier les différents systèmes neuronaux régissant cette miction. Il existe principalement deux voies neuronales; sensitives et motrices.

1.3.1 Les voies sensitives

Les différents récepteurs situés dans les viscères, transmettent leurs informations au système nerveux via des voies appelées afférentes; on parle donc ici d'information remontant de la périphérie vers les centres nerveux. Les sensations sont divisées en deux grandes catégories; extéroceptive et proprioceptive. La première regroupe toutes les sensations de douleur, de température et de toucher. La deuxième, quant à elle, regroupe les sensations d'étirement et de tension au niveau des tissus musculaires. La majorité des neurones sensitifs de la région vésicale projettent leurs axones dans la région sacrée de la moelle épinière via les nerfs érecteurs et les nerfs honteux externe. Une portion de ces neurones rejoint également la moelle au niveau thoraco-lombaire via les nerfs hypogastriques. Ces neurones afférents concernent principalement les propriétés extéroceptives du trigone et ne seront donc pas abordés.

1.3.2 Les voies motrices

Les influx nerveux contrôlant le système urinaire proviennent du SNC et empruntent les voies dites efférentes (motrices); on parle donc ici d'informations descendantes vers les organes. Les influx moteurs empruntent soit l'innervation végétative (autonome) ou l'innervation somatique. L'innervation végétative est celle qui règle le fonctionnement des viscères et est lui-même constitué de deux sous-systèmes; le sympathique et le parasympathique. L'innervation somatique régit quant à elle tout ce qui est « moteur volontaire » au niveau du corps (mouvements des membres par exemple).

Le détrusor serait innervé par des fibres parasympathiques via les nerfs pelviens tandis que le sphincter strié serait innervé par des fibres somatiques via les nerfs honteux à partir de S1 jusqu'à S4. Il est important de mentionner que ces conclusions ne font pas l'unanimité au sein de la communauté médicale et que plusieurs travaux arrivent à des conclusions différentes [7].

1.3.3 Anatomie d'un nerf

On peut constater que les nerfs en général sont constitués de plusieurs fascicules. Ces fascicules sont eux-mêmes constituées de plusieurs neurones possédant chacun un axone (Figure 1-2).

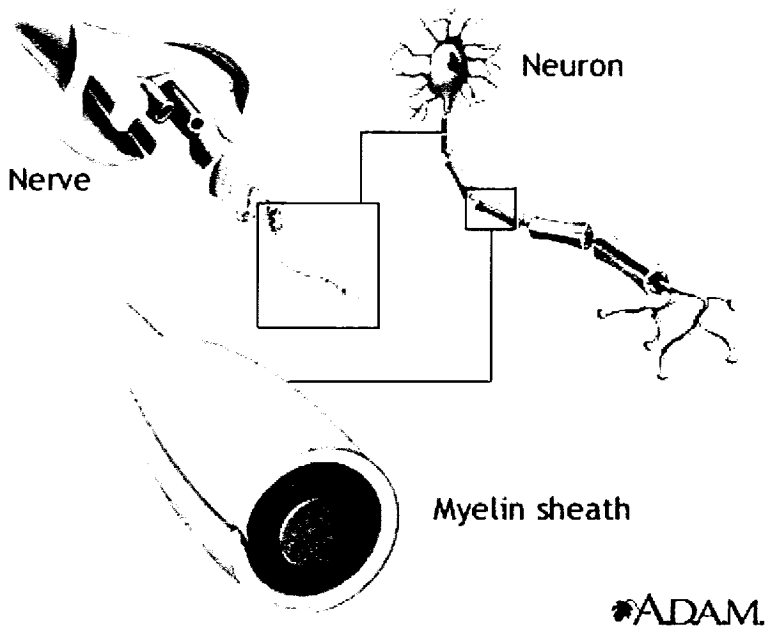


Figure 1-2: Anatomie d'un nerf (tiré de [1])

La Figure 1-3 montre une coupe transversale d'un nerf et les différents milieux entourant les axones et les fascicules. À l'intérieur d'un nerf, les fascicules sont regroupés par un tissu appelé epineurium. Chaque fascicule est entouré d'une mince couche (comparable à une enveloppe) appelée perineurium. À l'intérieur, les axones sont entourés par l'endoneurium et finalement, une couche de gras entoure le nerf à l'interface

avec l'intérieur du corps. Tous ces tissus présentent des caractéristiques différentes de conduction électrique qui seront abordées dans le chapitre 2.

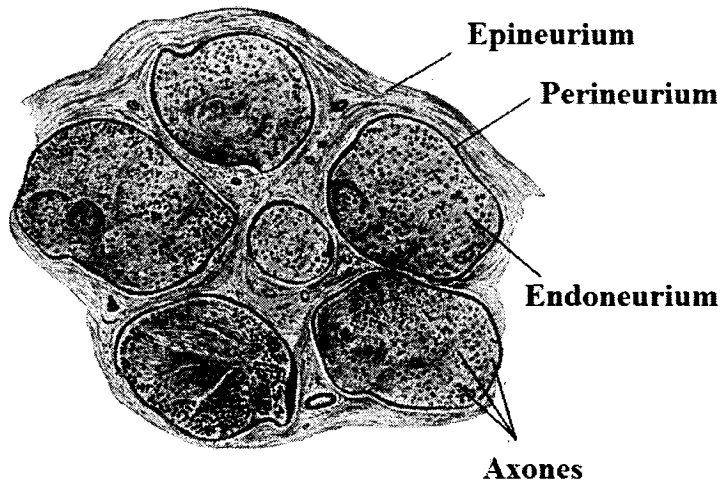


Figure 1-3 : Coupe transverse d'un nerf et ses différentes couches de tissus

Le cœur de la transmission des influx nerveux s'effectue via les axones; on les classes en deux grandes catégories : axones myélinisés ou non-myélinisés. Lorsque myélinisé, un axone est électriquement isolé sur la majorité de sa longueur et la propagation des potentiels d'actions (PA) est plus rapide (voir section 1.4.4). La Figure 1-4 montre une coupe d'un axone myélinisé; on y voit les couches de myéline qui entoure l'axone ainsi qu'un nœud de Ranvier (N). Les olygodendrocites (G) sont responsables de la formation de la myéline par l'enroulement successif de l'axone avec plusieurs couches.

1.3.4 Mécanisme de transmission de l'influx nerveux

Le SNC transmet et reçoit toutes les informations nécessaires au bon fonctionnement de toutes les parties du corps via les voix neuronales. Tel que mentionné précédemment, les voix sensitives agissent à titre de capteurs et transmettent les informations au SNC; ce dernier analyse ces informations et utilise les voies motrices pour réagir. Ces « informations » voyagent sous forme d'impulsions appelées potentiels d'action (PA).

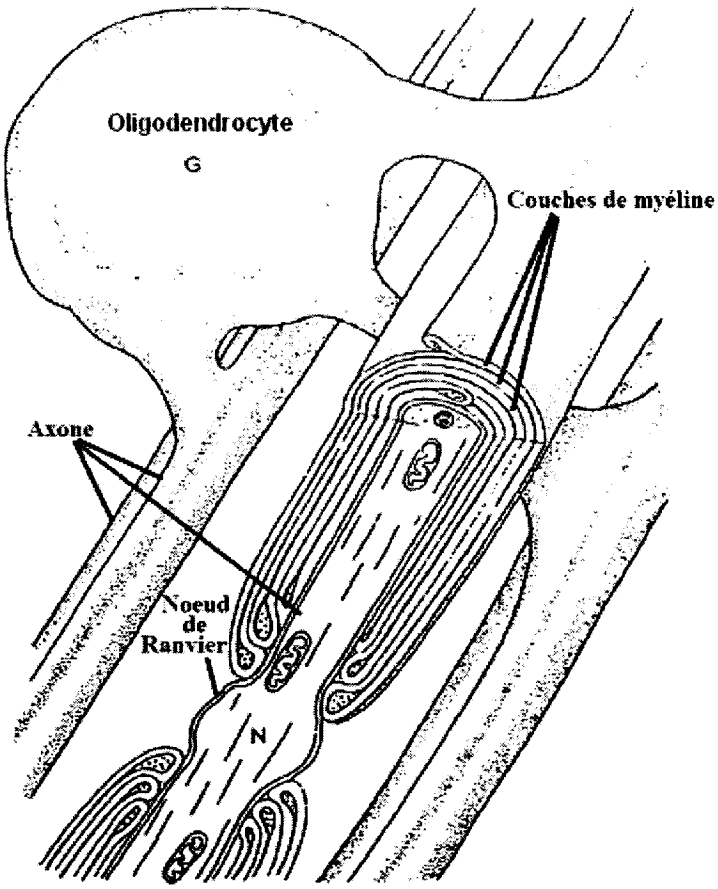


Figure 1-4 : axone myélinisé

Les PA ont été étudiés attentivement depuis une cinquantaine d'années, mais plus particulièrement par Hodgkin et Huxley [11]. Ces auteurs ont démontré que les PA déclenchés par une dépolarisation étaient constitués des phases suivantes (Figure 1-5): dépolarisation, repolarisation, hyperpolarisation et dépolarisation post-PA.

Ces auteurs ont également découvert qu'il existait plusieurs canaux permettant aux ions d'entrer et de sortir des axones engendrant ainsi plusieurs dépolarisations locales; la Figure 1-6 montre une portion d'un axone non myélinisé; un potentiel d'action vient

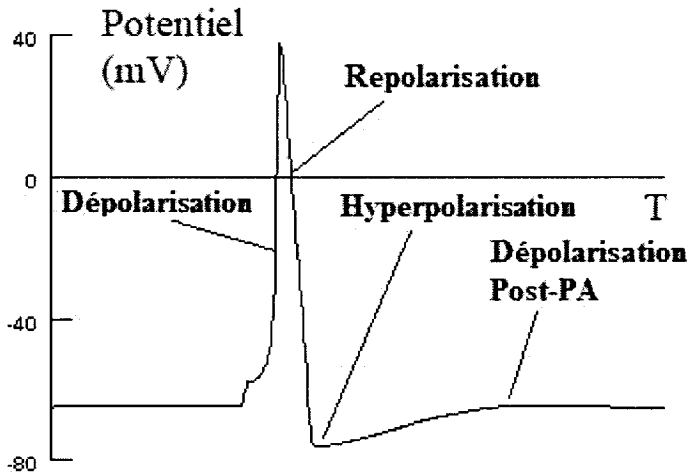


Figure 1-5 : Évolution du potentiel transmembranaire suivant une impulsion dépolarisante

d'être déclenché au site « B ». Un flux entrant d'ions de sodium (Na^+) dépolarise la membrane et engendre une boucle de diffusion locale des ions Na^+ de l'extérieur vers l'intérieur.

Cependant, au site « A » (où l'axone est en phase de repolarisation suite au passage d'un PA) les ions sortant de Na^+ s'additionnent aux ions sortant de Potassium (K^+). Cette migration des ions de sodium vers l'intérieur au site « B » engendre également une dépolarisation sous seuil au site « C ». Lorsqu'il sera suffisamment dépolarisé, le site « C » sera le prochain site de passage du PA et la membrane entrera en phase de repolarisation au site « B ». Ces « boucles » s'auto génèrent tout au long de l'axone permettant ainsi la propagation du PA.

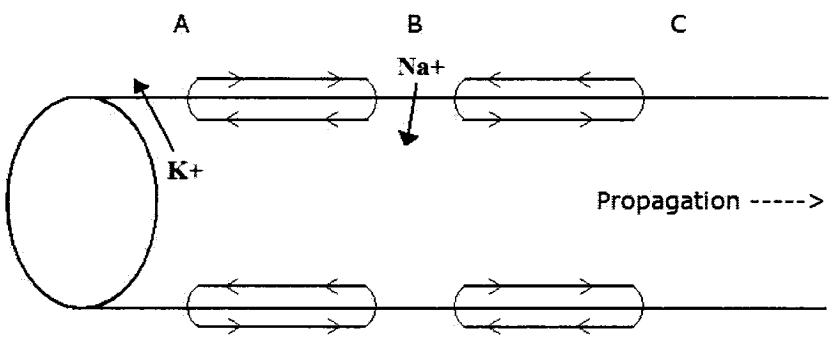


Figure 1-6 : Boucles de courant – Axone non myélinisée

Dans le cas des fibres myélinisées, le principe est exactement le même sauf que les « boucles » de courants se forment entre deux nœuds de Ranvier (Figure 1-7). La vitesse de propagation s'en trouve augmenté puisque les PA ne peuvent « apparaître » qu'aux nœuds de Ranvier; c'est ce qu'on appelle la propagation saltatoire.

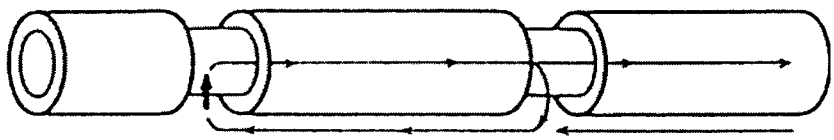


Figure 1-7 : Boucles de courant - Axone myélinisée

Il est important de mentionner que la gaine de myéline n'est pas un isolant parfait et qu'il existe des fuites d'ions entre l'axone et cette gaine [16].

1.4 Morphologie des voies motrices

Les voies motrices au niveau du système nerveux périphérique sont constituées de deux parties; le système végétatif (autonome) et le système somatique (volontaire). Le système somatique se charge des relations de l'organisme avec son environnement; récepteurs périphériques, organe des sens et le contrôle des muscles volontaires (striés). Le système végétatif est subdivisé en système sympathique et parasympathique qui ont tous deux la même organisation fondamentale : un neurone préganglionaire dont le corps cellulaire (soma) est dans le SNC et un neurone postganglionaire qui, après synapse avec le premier neurone dans un ganglion périphérique, rejoint les viscères auxquels il est destiné. Pour le système sympathique, les corps cellulaires des neurones se situent principalement entre les segments T1 et L3 (chez l'humain) de la moelle. Les neurones postganglionaires débutent à la synapse (ganglions) puis rejoignent les viscères appropriés.

Dans le système parasympathique, les corps cellulaires des neurones se situent majoritairement dans quelques noyaux du tronc cérébral et entre les segments S2 à S4.

La synapse avec le neurone postganglionaire est située dans des ganglions à proximité ou à l'intérieur des viscères innervés.

1.4.1 Physiologie et fonctionnement des neurones

Les neurones sont la base même de tout le système nerveux; un neurone est constitué de trois parties distinctive : les dendrites, le corps cellulaire et l'axone. Les dendrites supportent une arborescence de récepteurs synaptiques qui constituent le point d'entrée de l'information, cette information est ensuite dirigée vers le corps cellulaire (soma) qui agit à titre de centre intégrateur. Lorsque la dépolarisation atteint un seuil, il y déclenchement des PA qui se propagent jusqu'à l'extrémité de l'axone.

La géométrie des neurones varie selon leur fonctionnalité, ainsi, on différencie quatre types majeurs :

- Les **inter-neurones** qui servent à connecter deux neurones via une synapse.
- Les **récepteurs** qui captent les différentes informations provenant de l'extérieur et les transmettes au système nerveux central.
- Les **neurones** principaux constituant du système nerveux.
- Les **motoneurones** qui contrôlent le système moteur.

Normalement, l'excitation des synapses induit une variation dans le potentiel transmembranaire du corps cellulaire. Lorsque cette variation de potentiel est suffisamment élevée il y a déclenchement d'un PA. Dans le cas présent (la vessie), les PAs proviennent en partie des voies réflexes, mais principalement de la portion supra-médullaire du cortex. Suite à une lésion de la moelle, l'excitation provenant du cerveau ne peut se rendre jusqu'aux corps cellulaires en question. L'implant développé par notre équipe Polystim se connecte directement sur le nerf via une électrode et « joue » le rôle normalement effectué par le corps cellulaire. En injectant localement des charges via les électrodes, on arrive à faire varier suffisamment le potentiel transmembranaire de l'axone pour déclencher des PAs. Nous pouvons varier trois paramètres de stimulation afin

d'obtenir la meilleure réponse possible : l'amplitude du courant, la fréquence de stimulation et la largeur (temps) de l'impulsion. Grâce à ces trois paramètres, il est possible de produire une trame de stimulation jouant le même rôle que celle envoyée normalement par le corps cellulaire.

1.4.2 Potentiel transmembranaire

La membrane de l'axone est constituée d'une double couche lipidique dans laquelle sont insérés quelques protéines jouant le rôle de canaux pour certains ions. Les principaux ions liés à la propagation des PA sont le sodium (Na^+), le potassium (K^+), le calcium (Ca^{2+}) ainsi que le chlore (Cl^-). La membrane possède un potentiel dit de repos se situant aux alentours de -60mV (selon le type d'axone, ce potentiel de repos peut varier de -45mV à -75mV) et qui dépend du gradient de concentration du K^+. Un autre mécanisme entre alors en ligne de compte (nous n'aborderons pas ici les détails); il s'agit du mouvement des ions de sodium et de potassium entre le milieu extracellulaire et le milieu intracellulaire. Au repos, les ions de potassium peuvent facilement sortir ou entrer de la cellule alors que les ions de sodium ne le peuvent pas. La concentration des Na^+ étant plus grande à l'extérieur, les K^+ ont tendance à sortir; ainsi on assiste à une perte d'ions positifs de l'intérieur vers l'extérieur créant ainsi une différence de potentiel entre le milieu externe et le milieu interne. Pour les ions négatifs (Cl^-) le gradient de concentration à tendance à faire entrer ces anions dans la cellule alors que le potentiel transmembranaire négatif, les repousse. Ainsi, le flux d'anions au repos est très faible.

Une dépolarisation se produit lorsque les canaux de Na^+ s'ouvrent et qu'un flux important des ions de l'extérieur vers l'intérieur surgit afin de ramener la concentration d'ions de part et d'autre de la membrane au point d'équilibre. À l'inverse, une hyperpolarisation survient lorsque les canaux de K^+ (ou les canaux de Cl^-) s'ouvrent permettant ainsi la sortie d'ions K^+ vers l'extérieur (amenant ainsi d'avantage de charges négatives à l'intérieur de la membrane) et créant un potentiel plus négatif que celui du

repos. Un PA est initié lors d'une augmentation soudaine de la perméabilité de la membrane aux ions sodium.

1.4.3 Le potentiel d'action

Les PA sont déclenchées en présence d'une dépolarisation de la membrane habituellement provoquée par l'ouverture des canaux de sodium. Cette ouverture est obtenue grâce aux neurotransmetteurs, mais peut également être obtenue à l'aide s'une stimulation externe (électrique par exemple). L'ouverture de canaux de sodium se fait uniquement lorsqu'une certaine valeur seuil est dépassée. Une fois le seuil atteint, l'ouverture engendre un flux d'ions Na^+ vers l'intérieur de la membrane et par le fait même, une dépolarisation de celle-ci. Cette dépolarisation engendre l'ouverture d'encore plus de canaux de sodium permettant ainsi à plus d'ions d'entrer, créant un phénomène « boule de neige ». Une fois tous les canaux de sodium ouverts, le flux d'ions vers l'intérieur arrête lorsque le gradient de concentration est égal à la force électrique vers l'extérieur; i.e. lorsque le potentiel transmembranaire atteint +55mV. Une fois le pic atteint à +55mV, les canaux se ferment et les ions de sodium ne se déplacent plus. Le potentiel transmembranaire positif et le gradient de concentration force les ions K^+ à sortir de la cellule. Un flux d'ions positif vers l'extérieur rend alors l'intérieur moins positif (donc négatif); cette phase s'appelle re-polarisation. L'activation et l'inactivation des canaux de potassium sont plus lentes que pour les canaux de sodium; il en résulte une hyperpolarisation de la membrane après la re-polarisation. À ce moment, le potentiel transmembranaire est inférieur au potentiel de repos initial. (Figure 1-5).

Il est important de mentionner qu'une autre catégorie d'ions entre en ligne de compte dans les PA; il s'agit du calcium (Ca^{2+}). Son comportement est semblable aux ions de Na^+ dans le sens que la concentration externe est beaucoup plus élevée qu'à l'interne et qu'il passe au travers de la membrane via des canaux de calcium. Ils jouent donc un rôle mineur dans la phase de dépolarisation. Cependant, les canaux qui régulent le flux de

Ca^{2+} sont beaucoup plus lents que ceux régulant le flux de Na^{+}. Toutefois, son principal effet réside dans la concentration de calcium retrouvée à l'extérieur de la membrane; en effet, une accumulation ou un manque important de Ca^{2+} modifie l'excitabilité de la membrane; cela provoque un abaissement ou une augmentation du seuil de déclenchement d'un PA.

1.4.4 Codage fréquentiel des PA

Les neurones peuvent, dépendamment de l'intensité des stimuli qu'ils reçoivent, modifier la fréquence à laquelle ils génèrent des PA. Si un stimulus est assez fort et long pour induire une dépolarisation de la membrane et la conserver, certains neurones ne déchargeront pas un seul PA mais plusieurs tant et aussi longtemps que le stimulus est maintenu. Il est également possible qu'un neurone décharge des PA sous formes de train d'impulsions. On peut donc parler ici de « codage fréquentiel » des PA [6].

1.4.5 Période réfractaire

Il existe deux périodes réfractaires; une absolue et l'autre relative. Ces périodes imposent une limite supérieure à la fréquence à laquelle une cellule peut décharger. Les cellules ne peuvent conduire que dans un seul sens, cependant, en présence d'une stimulation artificielle (électrique par exemple) il est possible que les PA se propagent de part et d'autre du point de stimulation.

1.4.6 Conduction électrique des impulsions

Les propriétés physiques de l'axone jouent un rôle prédominant quant à la propagation des PA; en effet, en comparaison d'un conducteur de cuivre par exemple, l'axone est peu conducteur. La raison principale provient de la très grande résistivité du milieu interne de l'axone, il s'agit d'une solution relativement pauvre en électrolytes (très peu de particules

chargées). De plus, la taille de l'axone est généralement inférieure à 20 μm avec une impédance interne très élevé et la membrane possède des propriétés capacitives; ce qui engendre une diminution rapide de l'amplitude du courant qui voyage au sein de l'axone.

Toutes ces propriétés font en sorte que les PA qui se propagent le long de l'axone seraient immédiatement « éteints » après avoir parcouru une très petite distance. La couche de myéline entourant l'axone augmente considérablement la résistance de la membrane et diminue l'effet capacitif augmentant ainsi l'isolation électrique. La conduction des PA est plus rapide dans les axones myélinisés comparativement aux axones non-myélinisés.

1.4.7 Vitesse de conduction et intégration synaptique

La vitesse de conduction des axones dépend beaucoup de sa géométrie; plus l'axone possède un diamètre élevé, plus les PA se propageront rapidement (Tableau 1-1). Une règle du pouce simple est de multiplier par six le diamètre de l'axone; par exemple un axone de 20 μm de diamètre conduira à une vitesse approximative de 120 m/sec [6].

La sommation de toutes les impulsions (certaines excitatrices, certaines inhibitrices) reçues par le corps cellulaire est nécessaire afin de dépolariser la membrane jusqu'au seuil nécessaire pour déclencher un PA. Plus la somme des influx excitateurs sont forts, moins long sera le temps nécessaire pour dépolariser à nouveau la membrane et déclencher un deuxième PA. La fréquence de décharge des PA est l'expression de la somme de toutes les entrées synaptiques.

Prenons par exemple un motoneurone innervant des centaines de fibres musculaires striées; un tel neurone peut avoir aux environs de 50 000 synapses provenant de plusieurs endroits. Certaines synapses proviennent de senseurs informant le motoneurone sur le mouvement à effectuer, d'autres l'informent à propos de la position du corps dans l'espace, d'autres sur la vitesse avec laquelle le mouvement devrait être effectué, etc. La

somme des toutes ces entrées excitatrices et inhibitrices détermine la fréquence de décharge des PA et, par le fait même, de l'amplitude de contraction du muscle.

1.5 Classification des différentes fibres nerveuses

Les fibres nerveuses ont été répertoriés et classés selon le Tableau 1-1.

Tableau 1-1 : Classification des axones présents dans les nerfs périphériques (tiré de [14])

Type de Fibre	Diamètre (µm)	Vélocité de conduction (m/s)	Fonction
A-alpha	13 – 22	70 – 120	Motoneurones alpha, fuseau neuromusculaire primaire, organes de tendons de Golgi, toucher
A-beta	8 –13	40 – 70	Toucher, cinesthésie, fuseau neuromusculaire secondaire
A-gamma	4 – 8	15 – 40	Touché, pression, motoneurones gamma
A-delta	1 – 4	5 – 15	Douleur, contact brut, pression, température
B	1 – 3	3 – 14	Pré-ganglionaire autonome
C	0.1 – 1	0.2 – 2	Douleur, touché, pression, température, post-ganglionnaire autonome

Cette classification est basée sur le diamètre des fibres et sur la vitesse de conduction; en effet, plus la fibre possède un diamètre élevé, plus les PA se propagent rapidement à l'intérieur de celle-ci. En prenant une section d'un nerf et en appliquant un stimulus à une extrémité il est possible de mesurer le décalage entre la propagation des PA dû à la vitesse de conduction. En augmentant peu à peu l'amplitude du stimulus, on mesure après un certain temps la réponse des fibres plus petites (Figure 1-8) Il est important de noter que même si une grosse fibre peut propager des PA à une grande vitesse, sa période réfractaire est généralement plus élevée qu'une petite fibre.

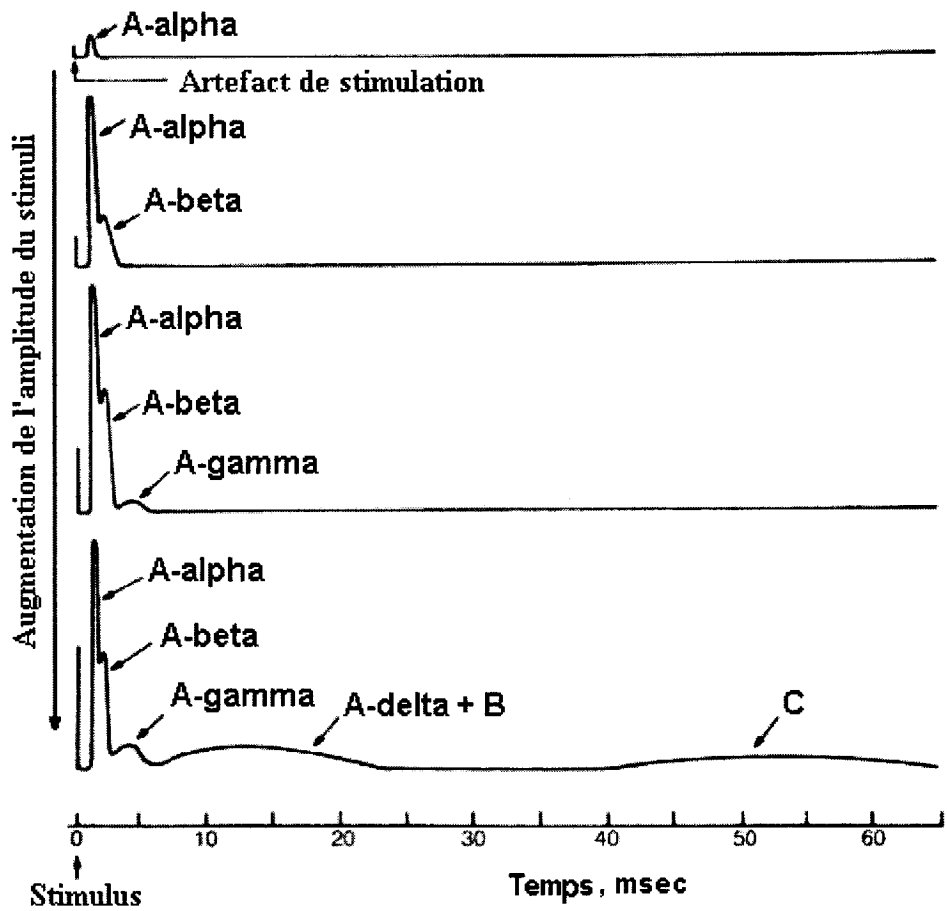


Figure 1-8 : Expérience sur la vitesse de conduction en fonction du diamètre (tirée de [14])

Il existe deux périodes réfractaires; une absolue et une relative. La période absolue est celle qui suit immédiatement après la polarisation; tout au long de cette période, il est impossible de déclencher un deuxième PA. Ensuite, vient la période relative; au cours de celle-ci, il est possible de déclencher un second PA si on augmente l'amplitude et la durée du stimulus. Ce sont ces périodes qui limitent la capacité d'un axone à déclencher des PA à des fréquences élevées. Cependant, la fréquence de stimulation à laquelle les fibres musculaires peuvent répondre est beaucoup plus faible. C'est donc dire que la période réfractaire musculaire est beaucoup plus longue que celle des fibres nerveuses.

1.6 Techniques de stimulation des racines sacrées

Nous pouvons classer actuellement les méthodes de stimulation pour surmonter le problème de dyssynergie et permettre la vidange de la vessie en 4 principales classes : la vidange post-stimulus, le blocage anodique, la fatigue du sphincter et la stimulation sélective par blocage haute fréquence. Le Tableau 1-2 résume les différentes classes et montre les avantages ainsi que les inconvénients de chacune.

1.6.1 Vidange post-stimulus

Cette technique est basée sur le temps de relaxation (période réfractaire) des fibres musculaires striées constituant le sphincter externe versus le temps de relaxation des fibres musculaires lisse constituant le détrusor. Les fibres lisses (blanches) se relâchent moins rapidement que les fibres striées (rouges); en utilisant un certain patron de stimulation appliqué sur les racines sacrées, on obtient une vidange saccadée de la vessie. Cette technique est la plus utilisée aujourd'hui et un neurostimulateur développé par Brindley a été commercialisé et présente de bons résultats cliniques [5].

1.6.2 Blocage anodique

Cette technique permet de palier au problème de dyssynergie en activant de manière sélective les fibres parasympathiques en combinant la stimulation de toute les fibres à un blocage anodique sélectif [21]. Le courant nécessaire au blocage des PA se propageant dans les fibres somatiques est inférieur à celui nécessaire pour bloquer les PA des fibres parasympathiques, une hyperpolarisation de la membrane à l'anode d'une électrode tripolaire permet de bloquer les PA se dirigeant vers le sphincter strié tout en laissant passer ceux qui se propage vers le détrusor. Ce blocage anodique permet de réduire de manière significative (plus de 80%) la pression présente à l'urètre en présence de stimuli de courant.

1.6.3 Fatigabilité du sphincter

Les fibres musculaires striées sont moins résistantes à une contraction de longue durée; elles sont enclines à produire une contraction forte mais de courte durée contrairement aux fibres lisses qui produisent une contraction longue et soutenue. En stimulant les nerfs honteux avec un train d'impulsion haute fréquence tout en maintenant une stimulation basse fréquence sur les nerfs sacrés, on arrive à fatiguer le sphincter externe [13] et obtenir un relâchement de celui-ci. Des études en phases aigue et chronique sur des chiens ont démontré l'efficacité de cette méthode. Les résultats obtenus sont comparables à une rhizotomie des nerfs honteux, mais cette technique est beaucoup moins invasive et elle est réversible.

1.6.4 La stimulation sélective

La stimulation sélective introduite par notre équipe Polystim est composé de deux trains d'impulsions bipolaires superposés [24] [4] et consiste à exciter les racines sacrées (innervant à la fois le sphincter externe et le détrusor). Le premier train d'impulsions basse fréquence possède une amplitude plus élevée et provoque la contraction du détrusor. Le second train haute fréquence d'amplitude plus faible inhibe la contraction du sphincter. Un prototype de ce stimulateur a été réalisé par Robin et al. [22] et à permis de valider la technique lors d'essais *in-vivo.*

Les fibres somatiques innervant le sphincter externe possèdent un seuil d'activation plus bas que les fibres parasympathiques innervant le détrusor. Le principe est le suivant; il faut choisir une amplitude de stimulation pour les basses fréquences suffisante pour activer les fibres parasympathiques, mais une amplitude pour les hautes fréquences non suffisante pour activer les fibres somatiques. Ensuite, on choisit une fréquence de stimulation (haute fréquence) suffisamment élevé pour inhiber le déclenchement de PA. On provoque ainsi la contraction du détrusor tout en obtenant un relâchement du

sphincter externe, permettant à l'urine d'être expulsée. Le présent mémoire propose un modèle du nerf validant cette théorie.

Tableau 1-2 : Résumé des techniques de stimulation des racines sacrées dédiées à la vidange de la vessie (Tiré de [4])

Technique	Avantages & Inconvénients	État
Vidange poststimulus	(+) 1 site d'implantation; (-) Pression intravésicale élevée (-) Miction saccadée	Utilisation clinique
Blocage anodique	(+) 1 site d'implantation	Essais en phase aiguë
Fatigue du sphincter	(+) Faible dyssynergie (-) 2 sites d'implantation	Essais en phase aiguë Technique obsolète
Stimulation sélective	(+) Faible dyssynergie (+) 1 site d'implantation	Essais en phase chronique

1.7 Conclusion

Ce premier chapitre présente des notions de base nécessaires à la compréhension des travaux présentés. Une connaissance approfondie des mécanismes entourant le déclenchement et la propagation des potentiels d'action facilite la réalisation du modèle présenté dans ce mémoire.

Chapitre 2

MODÉLISATION NEURONALE : PRINCIPAUX TRAVAUX

2.1 Modélisation neuronale

C'est en 1952 que Hodgkin et Huxley (HH) ont décrit pour la première fois toute la dynamique de la membrane d'une cellule nerveuse à l'aide d'expériences réalisées sur un axone géant de calmar. Par la suite, en 1964, Frankenhaeuser et Huxley (FH) ont décrit les équations régissant le potentiel transmembranaire d'un axone myélinisé à partir d'expériences effectuées sur un nerf de grenouille. Dans les années suivantes, les nerfs myélinisés des mammifères ont été modélisés à partir d'expérience sur les lapins et les chats. C'est en 1976 que McNeal [17] propose le premier modèle constitué de plusieurs segments représentés indépendamment par les équations de FH et connectés ensemble via une résistance axoplasmique. En 1987, Sweeney et al. [26] ont publié le premier modèle pour des mammifères à sang chaud basé sur les données obtenues par Chiu et al.[8] Depuis, plusieurs équipes ont travaillé sur des modèles plus ou moins complexes afin de simuler et de prédire le comportement des nerfs en présence d'une stimulation électrique. Voici quelques modèles présentés au cours des deux dernières décennies.

En 1989, Veltink [28] présentait un modèle permettant de déterminer le recrutement des fascicules à l'intérieur de 2 nerfs suite à une stimulation électrique. L'idée était de positionner une électrode et de calculer son champ d'action afin de vérifier quel(s) fascicule(s) pourrait être stimulé(s). Son modèle 3D consistait en un volume d'éléments finis représentés par des triangles équilatéraux dans le plan x-y et de pyramide triangulaire dans le plan z (Figure 2-1)

Cet auteur a présenté plusieurs résultats démontrant que les électrodes externes n'arrivaient pas à recruter sélectivement les fascicules ; principalement ceux situés au

centre du nerf. Sa conclusion était que l'électrode intra fasciculaire mériterait beaucoup plus d'attention et pourrait être une solution pour l'avenir.

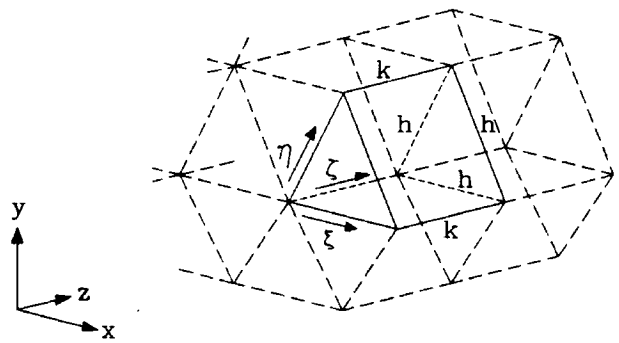


Figure 2-1 : Grillage d'éléments finis selon le modèle réalisé par Veltink (tirée de [28])

En 1993, Frank Rattay [19] publie une revue de plusieurs modèles. Son article porte sur les modèles suivants : HH, FH, Sweeney et al., ainsi que celui proposé par Schwarz et Eikhof. L'article explique les différences entre chacun de ces modèles et en démontre les points forts et faibles. Il présente également côte à côte des résultats obtenus avec chacun d'eux. La conclusion fait mention que le modèle classique de HH est le meilleur pour simuler la réponse du nerf auditif, mais que les autres modèles sont mieux pour décrire les réponses des motoneurones du corps humains (en prenant compte du facteur de correction de température dans le cas du modèle FH). Les modèles qui suivirent cette revue sont de plus en plus complexes et tentent de prendre en compte le plus d'éléments extérieurs possibles.

En 1994, Rijckhoff [20] présente un modèle pour décrire la stimulation sélective par blocage anodique des racines sacrées afin de contrôler la vessie. Son modèle est constitué d'un cylindre représentant le nerf et de plusieurs couches successives représentant dans l'ordre, l'isolant autour de l'électrode, le métal de l'électrode, le fluide cérébrospinal dans lequel « baigne » l'électrode et le nerf et finalement, une surface représentant les tissues plus ou moins éloignés, compressés en une couche de 1 mm.

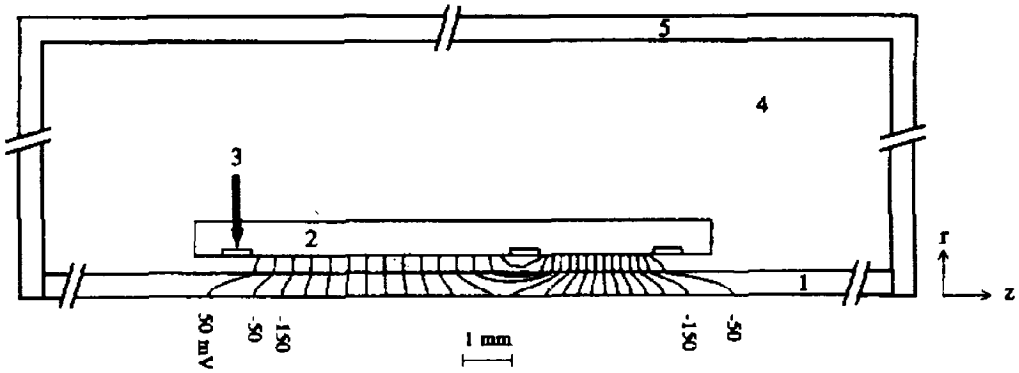


Figure 2-2 : Modèle d'un nerf et du milieu l'entourant
(1) Le nerf - (2) Isolant de l'électrode – (3) Contact de l'électrode (4) Fluide cérébrospinal – (5) Surface représentant tous les tissus plus ou moins éloignés regroupés en une seule couche de 1 mm (tiré de [20])

Les compartiments 2 à 5 de la Figure 2-2 possèdent une conductivité isotrope alors que le compartiment 1 (le nerf) possède une conductivité anisotrope et est modélisé selon les travaux réalisés par McNeal [17] et par Chiu [8].

Ce modèle a permis de mieux comprendre la technique du blocage anodique et d'en déterminer les paramètres optimaux de stimulation. Par le fait même, il a permis d'obtenir certaines informations concernant la géométrie des électrodes utilisées ; espacement idéal entre cathode et anode afin d'obtenir le meilleur blocage possible, etc.

Plus récemment (2000). Perez-Orive et Durand [18] ont publié des résultats concernant l'utilisation d'une forme aplatie d'électrodes permettant une plus grande sélectivité. Ils présentent les résultats obtenus en comparant une électrode ronde classique et une électrode aplatie (Figure 2-3). Ces auteurs parlent d'un index de sélectivité des fascicules à l'intérieur du nerf, des différences d'amplitude de stimulation nécessaires pour déclencher des PA ainsi que des données concernant la longueur de ce type d'électrodes. Il s'agit d'un modèle de type câble réalisé avec la méthode des éléments finis, de plus, l'axone est constitué de plusieurs compartiments suivant la dynamique des membranes décrites par Sweeney.

La conclusion des travaux démontre que l'utilisation d'une électrode de forme aplatie versus une électrode circulaire conventionnelle permet d'obtenir un indice de sélectivité beaucoup plus élevé et une amplitude de courant de stimulation moins élevée.

En 2002, McIntyre et al. ont publié un nouveau modèle [16]. Il s'agit d'un modèle détaillé d'un axone destiné à mieux comprendre la phase post-PA (ou phase de recouvrement). En effet, McIntyre et al. proposent la division du canal de sodium (tel que décrit précédemment par FH) en deux canaux distincts; un premier dit persistant et un deuxième rapide. Les résultats obtenus à l'aide du modèle reproduisent fidèlement plusieurs résultats expérimentaux obtenus par différentes équipes indépendantes. La morphologie et la dynamique des membranes utilisées dans ce modèle sont basées sur des expériences effectuées chez les humains, les chats et les rats.

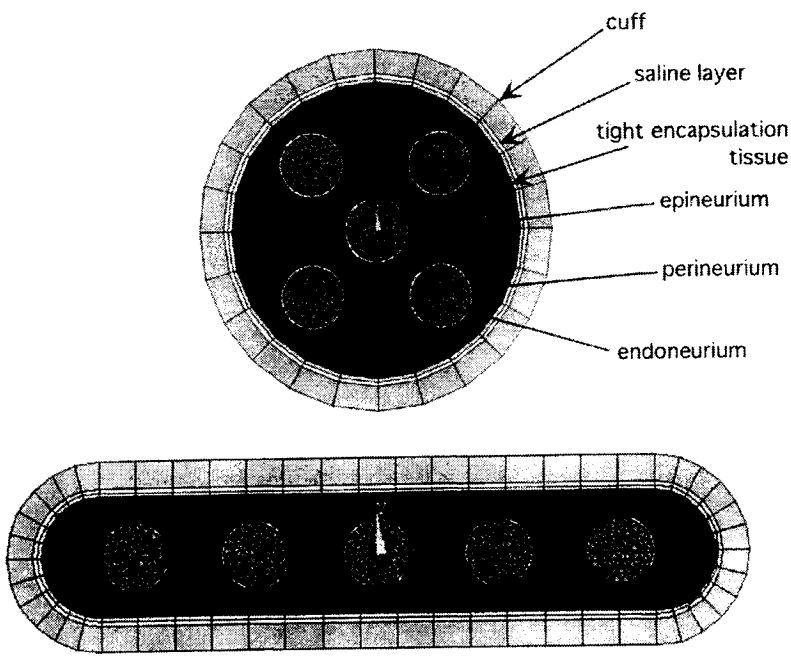


Figure 2-3 : Électrode classique de forme ronde et électrode aplatie proposée par Perez-Orive et Durand (tiré de [18])

Une structure doubles câbles imbriqués avec représentation explicite des nœuds de Ranvier, des sections internodale et paranodale ainsi qu'une impédance finie pour la

couche de myéline ont été utilisées (Figure 2-4). Ce modèle sera décrit plus précisément dans la prochaine section car il est à la base du modèle du nerf présenté dans cet ouvrage.

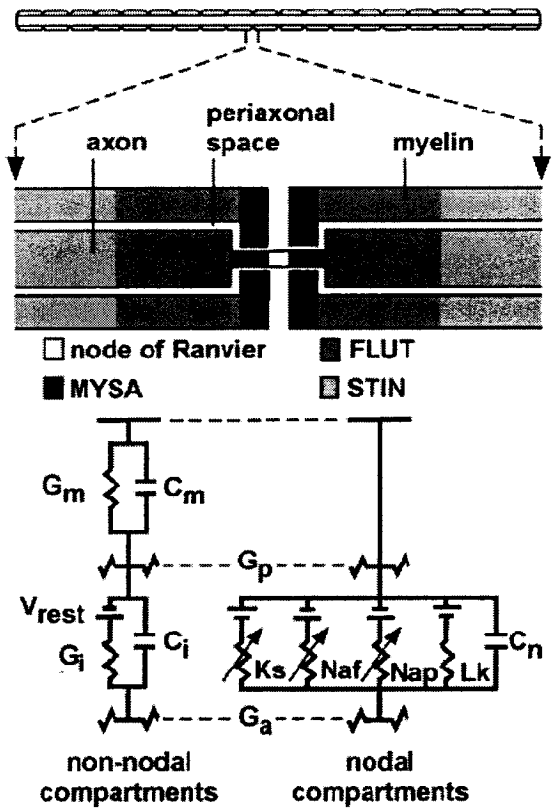


Figure 2-4 : Structure du modèle proposé par McIntyre et al. (tirée de [16])

La même équipe a publié un modèle plus complet incluant une modélisation de la moelle épinière (avec des éléments finis) [15]. Le modèle permet de positionner une électrode à l'intérieur de la moelle et de simuler son effet sur les neurones environnants (Figure 2-5). Les dendrites et les axones des neurones sont modélisés selon une structure double câbles imbriqués présentées ci-dessous. Les résultats obtenus ont démontré que l'utilisation d'un stimulus biphasique asymétrique permettait d'obtenir une activation sélective des neurones près de l'électrode ou des fibres provenant d'autres neurones lorsqu'on répétait le stimulus. La variation de la fréquence de stimulation permet également d'augmenter la sélectivité en faisant varier les différences d'excitabilité des corps cellulaires et des fibres après le passage d'un PA.

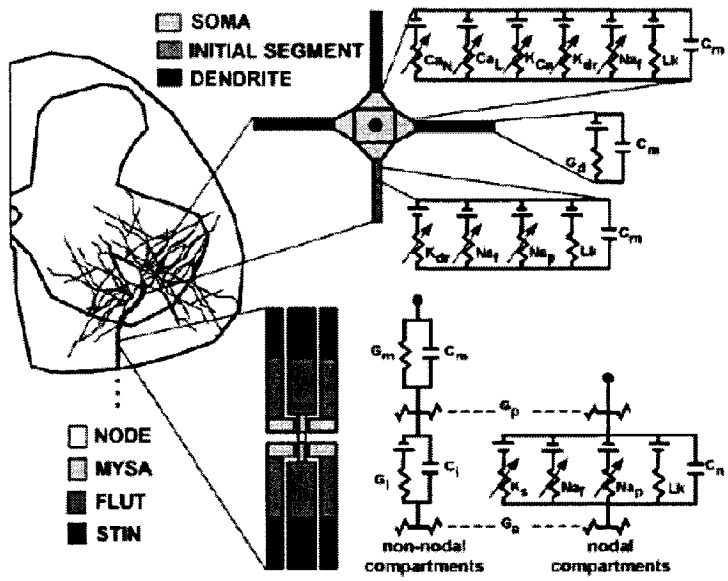


Figure 2-5 : Représentation de la moelle épinière et d'un axone se dirigeant vers la vessie (tiré de [15]).

2.2 Description du modèle de McIntyre et al.

Tel que présenté à la Figure 2-4 le modèle possède une structure double câbles superposés afin de modéliser explicitement la couche de myéline entourant l'axone. De plus, trois sections sont présentes afin de mieux représenter les différentes épaisseurs de la gaine de myéline lors que celle-ci se rapproche d'un nœud de Ranvier. En effet, l'épaisseur de la couche de myéline varie en fonction de sa position dans l'espace; au milieu du segment internodal, elle est à son plus épais alors qu'en s'approchant d'un nœud de Ranvier, elle s'amincie progressivement. Ces différentes sections sont représentées comme suit; « stereotype internodal region (STIN) » est la plus grande des trois sections représentant l'espace internodal. La « flutted internodal region (FLUT) » représente quant à elle, la portion où la gaine de myéline commence à s'amincir. La « myelin sheath attachment region (MYSA) » représente la portion où la myéline s'attache à l'axone juste avant le nœud de Ranvier. Ces trois sections sont définies comme étant passive dans la propagation des PA alors que les nœuds de Ranvier sont définis comme étant la région active.

Tel qu'illustrée à la Figure 2-4, toutes ces sections sont représentées par un circuit simplifié. La couche de myéline est modélisée par une conductance (G_m) et une capacité (C_m) en parallèle alors que les sections autres que les nœuds, sont modélisées par une conductance (G_i) en série avec un source de tension (V_{rest}), le tout en parallèle avec une capacité (C_i). G_p représente la conductance de l'espace située entre l'axone et la couche de myéline, G_a représente tout simplement la conductance de l'axolemme (intérieur de l'axone). Tel que mentionné précédemment, les nœuds de Ranvier constituent la région active lors de la propagation des PA, leur modélisation est donc un peu plus complexe. Toujours en rapport à la Figure 2-4, les différents canaux ioniques sont modélisés par une conductance variable (Na_p, Na_f et Ks) en série avec une source de tension (représentant le potentiel transmembranaire de repos de chacun des canaux). Une autre branche constituée d'une conductance fixe (Lk) et d'une source de potentiel modélise le phénomène de courant de fuite. Ces quatre branches sont en parallèles avec une capacité (Cn) qui modélise l'effet capacitif de la membrane de l'axone. Tel que mentionné précédemment, il existe d'autres canaux présents au niveau de la membrane, mais qui ne sont pas modélisés ici. Le Tableau 2-1 présente les paramètres électriques associés à chacune des composantes du modèle.

2.2.1 Équations du modèle

Afin de déterminer la valeur du potentiel transmembranaire de l'axone à chaque moment et ainsi simuler la propagation des PA, il faut évidemment solutionner des équations à différents degrés. Les équations utilisées dans ce modèle, sont basées sur le principe de fonctionnement des canaux ioniques décrit par HH. Tous les courants ioniques du modèle peuvent être écrits sous la forme générale suivante :

$$I_{ion} = g_{ion}(V_m - E_{ion})$$

Équation 1 : Forme générale de l'équation des courants ioniques

Tableau 2-1 : Paramètres électriques du modèle (adaptée de [16])

Variable	Symbole	Valeur
Capacité nodale	C_n	2 μF/cm^2
Capacité internodale	C_i	2 μF/cm^2
Capacité de la myéline	C_m	0.1 μF/cm^2
Résistivité axoplasmique	ρ_a	70 Ω cm
Résistivité périaxonale	ρ_p	70 Ω cm
Conductance de la myéline	g_m	0.001 S/cm^2
Conductance MYSA	g_a	0.001 S/cm^2
Conductance FLUT	g_f	0.0001 S/cm^2
Conductance STIN	g_i	0.0001 S/cm^2
Conductance max Na_f	g_{Naf}	3.0 S/cm^2
Conductance max Na_p	g_{Nap}	0.01 S/cm^2
Conductance max Ks	g_{Ks}	0.08 S/cm^2
Conductance de fuite nodale	g_{Lk}	0.007 S/cm^2
Potentiel de Nernst Na^+	E_{Na}	50.0 mV
Potentiel de Nernst K^+	E_K	-90.0 mV
Potentiel réversible de fuite	E_{Lk}	-90.0 mV
Potentiel de repos	V_{rest}	-80.0 mV

Dans laquelle g_{ion} est la conductance maximum pour chacun des canaux individuels, multipliées par une variable dite de « gating » (m, h, p et s) comprise entre 0 et 1 (Hodgkin and Huxley 1952).

La dépendance en temps et en tension de chacune de ces variables de « gating » est définie comme suit :

$$T_\omega = \frac{1}{\alpha_\omega + \beta_\omega}$$

$$\frac{d\omega}{dt} = \alpha_\omega(1-\omega) - \beta_\omega\omega$$

Équation 2 : Dépendance en temps et en tension des variable « gating »

Chacune des régions décrites comporte également des paramètres morphologiques. Le tableau suivant dresse une liste de ces paramètres en fonction du diamètre de la fibre pour chacune des sections du modèle.

Tableau 2-2 : Paramètres morphologiques (µm) des axones en fonction de leur diamètre (adapté de [16])

	Diamètres des fibres (µm)								
	5,7	***7,3***	***8,7***	***10,0***	***11,5***	***12,8***	***14,0***	***15,0***	***16,0***
Séparation nœud à noeud	500	750	1000	1150	1250	1350	1400	1450	1500
# de lamelles de myéline	80	100	110	120	130	135	140	145	150
Longueur Nœud	1	1	1	1	1	1	1	1	1
Diamètre Nœud	1,9	2,4	2,8	3,3	3,7	4,2	4,7	5,0	5,5
Longueur MYSA	3	3	3	3	3	3	3	3	3
Diamètre MYSA	1,9	2,4	2,8	3,3	3,7	4,2	4,7	5,0	5,5
Espace périaxonale MYSA	0,002	0,002	0,002	0,002	0,002	0,002	0,002	0,002	0,002
Longueur FLUT	35	38	40	46	50	54	56	58	60
Diamètre FLUT	3,4	4,6	5,8	6,9	8,1	9,2	10,4	11,5	12,7
Espace périaxonale FLUT	0,004	0,004	0,004	0,004	0,004	0,004	0,004	0,004	0,004
Longueur STIN	70,5	111,2	152,2	175,2	190,5	205,8	213,5	221,2	228,8
Diamètre STIN	3,4	4,6	5,8	6,9	8,1	9,2	10,4	11,5	12,7
Espace périaxonale STIN	0,004	0,004	0,004	0,004	0,004	0,004	0,004	0,004	0,004

Il faut donc déterminer les variables α et β pour chacun des canaux de conduction ionique. Ce modèle présente 4 différents canaux; courant de sodium rapide et lent, de potassium et de fuite :

$$I_{Na_f} = g_{Na_f} \times m^3 h \times (V_m - E_{Na})$$

$$\alpha_m = \frac{[6,57 \times (V_m + 20,4)]}{1 - e^{\frac{-(V_m+20,4)}{10,3}}}$$

$$\beta_m = \frac{[0,304 \times (-V_m - 25,7)]}{1 - e^{\frac{(V_m+25,7)}{9,16}}}$$

$$\alpha_h = \frac{[0,34 \times (-V_m - 114)]}{1 - e^{\frac{(V_m+114)}{11}}}$$

$$\beta_h = \frac{12,6}{1 + e^{\frac{-(V_m+31,8)}{13,4}}}$$

Équation 3 : Équations de courant ionique pour le canal de sodium rapide

$$I_{Nap} = g_{Nap} \times p^3 \times (V_m - E_{Na})$$

$$\alpha_p = \frac{[0,0353 \times (V_m + 27)]}{1 - e^{\frac{-(V_m+27)}{10,2}}}$$

$$\beta_p = \frac{[0,000883 \times (-V_m - 34)]}{1 - e^{\frac{(V_m+34)}{10}}}$$

Équation 4 : Équations de courant ionique pour le canal de sodium lent

$$I_{K_s} = g_{K_s} \times s \times (V_m - E_K)$$

$$\alpha_s = \frac{0,3}{1 + e^{\frac{V_m+53}{-5}}}$$

$$\beta_s = \frac{0,03}{1 + e^{\frac{V_m+90}{-1}}}$$

Équation 5 : Équations de courant ionique pour le canal de potassium

$$I_{Lk} = g_{Lk} \times (V_m - E_{Lk})$$

Équation 6 : Équation du courant de fuite

De plus, un autre paramètre doit être pris en compte puisque les expériences réalisées pas HH étaient à température ambiante et que le dans le cas présent on veut modéliser la propagation des PA à l'intérieur du corps, il faut ajouter un facteur de correction de température.

$$Q_{10_1} = 2,2^{\frac{T-20}{10}}$$

$$Q_{10_2} = 2,9^{\frac{T-20}{10}}$$

$$Q_{10_3} = 3,0^{\frac{T-36}{10}}$$

Équation 7 : Facteurs de correction de température

2.3 Modèles en cours de réalisation

L'équipe de Grill est actuellement active au niveau de la modélisation neuronale [15] [16]. En effet, cette équipe est à l'origine des deux derniers modèles présentés à la fin de la section 2.1. Leurs travaux sont de plus en plus utilisés. Le modèle présenté en 2002 reproduit fidèlement la réponse normale d'un nerf à une stimulation électrique.

L'équipe de Durand a travaillé et travaille toujours à l'élaboration d'une électrode de forme aplatie (Flat Interface Nerve Electrode - FINE). À cet effet, plusieurs modélisations ont été nécessaires et sont encore sous développement. En effet, en 2005 lors de la conférence annuelle de la « International Functionnal Electrical Stimulation Society – IFESS », des travaux ont été présentés par des membres de cette équipe concernant la modélisation de leur électrode et de ces effets sur la stimulation électrique d'un nerf.

Les travaux de cette équipe utilisent le modèle proposé par McIntyre et al. et est appliqué à la modélisation du nerf fémoral chez l'humain [23]. Les simulations permettront de mieux comprendre la stimulation électrique en général, mais plus précisément de simuler l'activation sélective de fascicules à l'intérieur du nerf grâce à leur modèle « FINE ». L'équipe de Durand développe depuis quelques temps une méthode de stimulation sélective des fascicules grâce à la forme aplatie de leur électrode :

en obtenant une forme moins circulaire et plus longitudinale, on peut supposer que les fascicules ont tendances à s'aligner sur une même ligne (Figure 2-6). En déterminant la configuration optimale des contacts de l'électrode, on peut ainsi faire passer le courant entre différents contacts afin d'activer sélectivement les fascicules voulus.

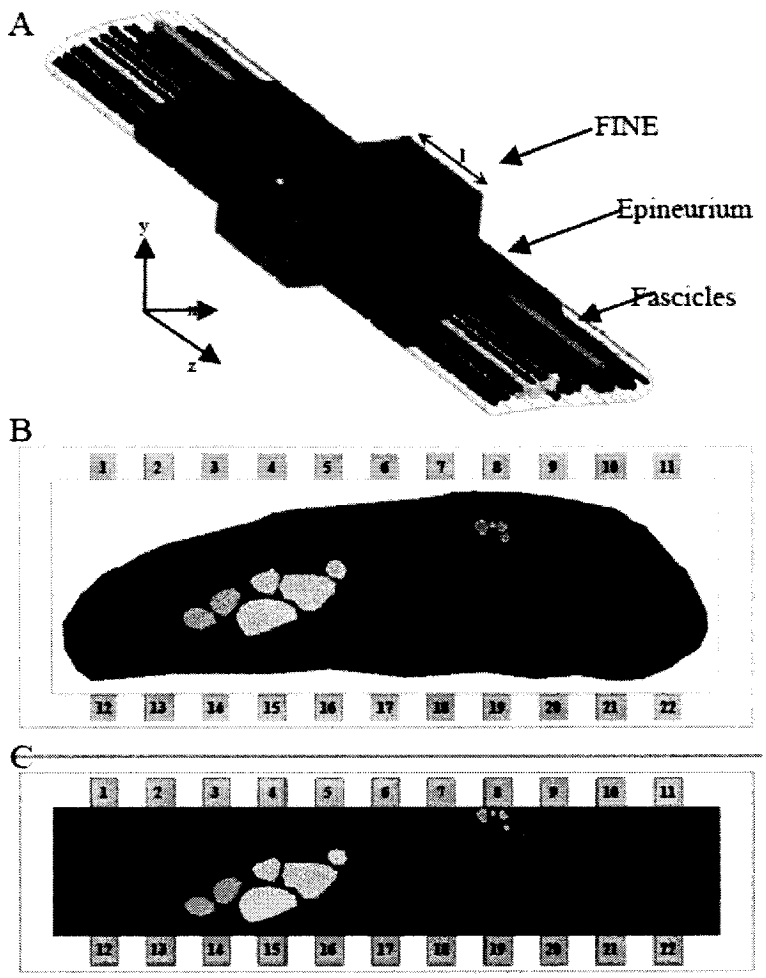


Figure 2-6 : A) Modèle du nerf avec "FINE" – B) Modèle de l'électrode avec une section du nerf fémoral - C) Modèle de l'électrode après remodelage du nerf (tiré de [23])

Les résultats obtenus ont démontré qu'il existait un optimum concernant la disposition des contacts tout autour de l'électrode « FINE » ainsi qu'une limite supérieure quant au nombre de ces contacts. Les résultats démontrent également que le remodelage du nerf à l'aide d'une électrode qui se referme après implantation augmente la sélectivité puisque la distance électrode - fascicule s'en retrouve diminuée. Cependant, les fascicules

innervant un même groupe de muscles sont habituellement regroupés, l'effet « d'aplatir » le nerf pourrait engendrer une redistribution de ces fascicules et le nombre de contacts nécessaires devraient alors être augmenté. Des études plus poussées sur la redistribution des fascicules après remodelage sont vivement recommandées.

L'équipe de Tai travaille également à l'élaboration d'un modèle d'axone non myélinisé basé sur les équations de HH [27]. Les travaux dans ce cas, portent sur le blocage de la conduction nerveuse à l'aide d'un train d'impulsions haute fréquence. Le modèle est constitué de plusieurs cylindres interconnectés comportant chacun une capacité et une résistance membranaire. Une résistance représentant la conductivité de l'axoplasme relie les cylindres entre eux. De plus, deux potentiels sont représentés, un intracellulaire et l'autre extracellulaire (Figure 2-7). Deux types d'électrodes ont été utilisés ; une première dites « électrode test » permettant d'induire le stimulus et l'autre, « électrode de blocage » permettant d'induire les impulsions de blocage hautes fréquences. Les simulations consistaient à induire une dépolarisation suffisante au niveau de « l'électrode test » (située à 10 mm) pour déclencher un PA et d'induire un train d'impulsions hautes fréquences agissant à titre de bloc au niveau de l'électrode de blocage (située à 25 mm). Les résultats obtenus démontrent que le blocage est en fonction de la fréquence ainsi qu'en fonction du diamètre des fibres que l'on tente de bloquer.

Les résultats démontrent également que le blocage des PA par un train d'impulsions haute fréquence est directement lié à la variable « n » des équations de HH ; en d'autres termes à l'activation des canaux de potassium. En présence du blocage, les variables d'action et de désactivation du sodium varient de la même manière que lorsqu'un PA est déclenché à ce site. Cependant, la variable d'activation du potassium atteint un niveau élevé et reste à ce niveau tant et aussi longtemps que le blocage est présent. De plus, il est démontré que pour toutes les fréquences, l'amplitude de stimulation nécessaire pour bloquer la conduction est inversement proportionnelle au diamètre de l'axone.

Finalement, le dernier modèle présenté ici est celui développé par notre équipe. Il est basé sur le modèle à structure double câble de McIntyre et al. tout comme le modèle de l'équipe de Durand. Cependant, notre modèle est utilisé afin d'améliorer la stimulation sélective développé dans le cadre du projet Urostim sur l'implant urinaire.

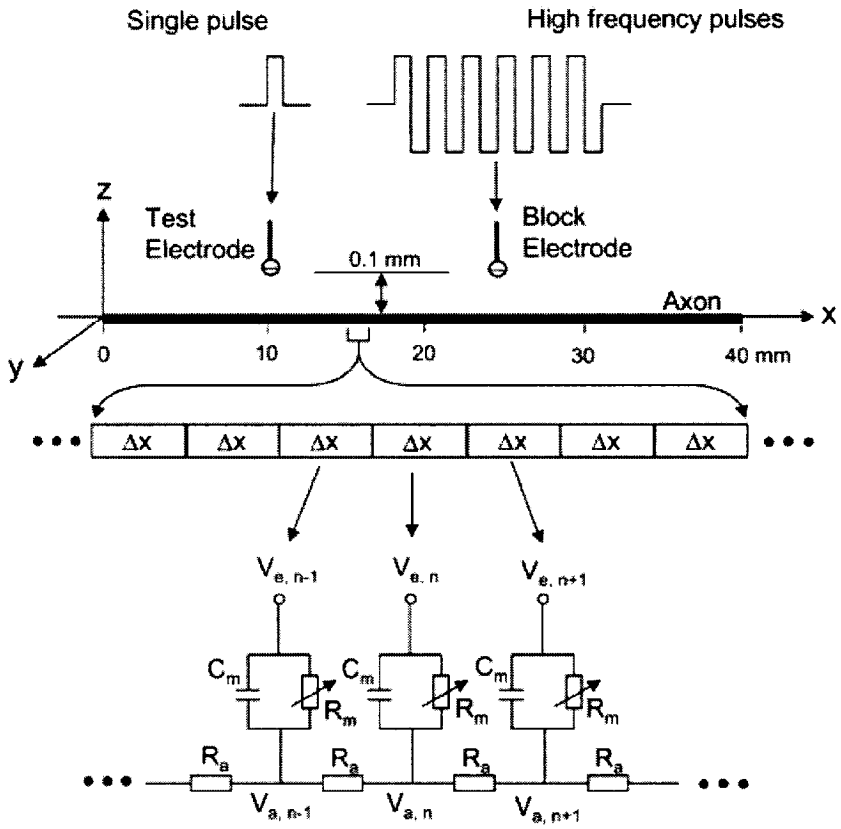


Figure 2-7 : Modèle d'axone non-myelinisé proposé par Tai et al. (tirée de [27])

2.4 Conclusion

Nous avons décrit les mécanismes régissant la propagation des PA au sein d'un nerf. Nous avons également présenté une brève revue de la littérature concernant les modèles présentés au cours des dernières années et ceux en cours de développement par quelques équipes travaillant dans le domaine de la neurostimulation. Le prochain chapitre est consacré à la description des principaux logiciels utilisés pour la réalisation du modèle que nous proposons.

Chapitre 3

LOGICIELS DE SIMULATION

3.1 Introduction

Dans le cadre des présents travaux, deux logiciels ont été utilisés pour le développement de notre modèle. Le premier permet la résolution des éléments finis et à été développé par l'université d'Utah; il s'agit de SCIRun/BioPSE [12]. Le second logiciel utilisé est Neuron [10]; il permet de simuler tous les mécanismes transmembranaires d'une cellule d'origine biologique. Il a été développé conjointement par l'université Duke et l'université de Yale. Le présent chapitre décrit ces logiciels, leur principe de fonctionnement et les raisons qui nous ont poussé à les utiliser.

3.2 Logiciel d'éléments finis (SCIRun/BioPSE)

Après consultation de la littérature à ce sujet, la décision d'utiliser la méthode des éléments finis afin de simuler l'effet du courant provenant des électrodes sur les différentes couches de tissus biologiques présentes dans le nerf semblait être le meilleur choix. Plusieurs logiciels d'éléments finis sont disponibles sur le marché (SolidWorks, ANSYS, etc.) cependant, ces logiciels, malgré leur renommé, servent habituellement à déterminer et à mesurer des forces provenant de contraintes physiques sur des pièces composés de différents alliages. Ils sont largement utilisés lors du processus de conception de pièces automobiles, de structures tel les ponts, etc.

Certaines de ces compagnies ont tout de même développé des modules permettant de simuler la propagation et la force des champs magnétiques et/ou électriques provenant de différentes sources. Il existe également des modules concernant l'écoulement des fluides, les différentes contraintes thermales, etc. Cependant, très peu de ces logiciels présentent

des modules permettant de simuler des environnements biologiques. Les données concernant les tissus biologiques sont plutôt rares et les recherches dans ce domaine sont plutôt récentes. Ces facteurs expliquent pourquoi peu de compagnies ont développé de tels modules.

Pour faire face à ce manque, le « Scientific Computing and Imaging Institute (SCI Institute) » de l'université d'Utah a entrepris de développer un logiciel utilisant les éléments finis dédié aux environnements biologiques. Récemment, ils ont étendus le champ d'action du logiciel à différents problèmes tel que l'exploration souterraine du pétrole et du gaz et à la façon dont se propage le feu (Figure 3-1).

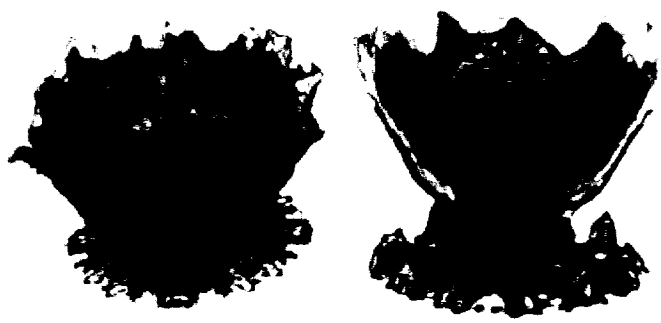


Figure 3-1 : Simulation de propagation du feu réalisé avec SCIRun (tirée de [12])

À l'origine, le logiciel a été développé pour modéliser des applications biomédicales tel que le problème « inverse » en cardiologie (Figure 3-2) ou la localisation de foyers épileptiques par exemple.

Le choix s'est donc porté sur ce logiciel pour les raisons suivantes :

1 Tel que mentionné précédemment, SCIRun, comparativement aux autres logiciels d'éléments finis, est destiné au traitement de données biologiques.

2 SCIRun est un logiciel développé dans un milieu universitaire et il est gratuit. Il s'installe aisément avec le système d'exploitation Linux et il s'agit d'un logiciel « open source ».

3 SCIRun offre un environnement de résolution de problème modulaire. Le logiciel est muni de plusieurs modules effectuant des tâches différentes sur les données.

Figure 3-2 : Champs bioélectriques produit par le coeur dans un torse humain (tiré de [12])

Chacun des modules de SCIRun prend des données sur un port d'entrée, effectue une opération et en ressort les résultats sur un port de sortie. On peut ainsi cascader plusieurs blocs afin de réaliser des opérations plus complexes. En plus de posséder une vaste gamme de blocs prédéfinis, SCIRun permet la création de bloc sur mesure; on peut créer soi-même un bloc en programmant les opérations que nous désirons lui faire faire. Dans le cas présent, un assemblage inclus dans SCIRun nous permettra d'effectuer toutes les opérations nécessaires; il s'agit du module BioPSE (Biological Problem Solving Environment). Ces modules seront décrits un peu plus loin.

Malgré le fait que SCIRun soit encore en développement, un grand nombre de travaux publiés font état de l'utilisation de SCIRun concernant la simulation de problèmes bio-électriques. Nous allons maintenant décrire les blocs présents dans SCIRun et dans le module BioPSE dont nous aurons besoin.

3.2.1 Modules SCIRun

SCIRun possède un agencement de blocs prédéfinis qui permettent de réaliser des opérations mathématiques de base ainsi que le traitement primaire de données. Mais, il possède également le nécessaire afin de réaliser l'affichage tridimensionnel des résultats de simulation. Pour ce qui est des fonctions plus complexes, SCIRun utilise les différents modules ajoutés (tel que BioPSE) ou encore les blocs définis par l'utilisateur. Dans le présent modèle, le module BioPSE sera utilisé afin de modéliser l'insertion de sources de tension ou de courant ainsi que pour tout ce qui touche la conductance des tissus. Les modules SCIRun seront utilisés afin de réaliser les opérations de base.

Premièrement, SCIRun traite les données géométriques comme des champs; on peut greffer beaucoup d'informations importantes relatives aux données géométriques à l'intérieur d'un champ. Par exemple, un champ peut comporter les coordonnées géométriques d'un maillage, mais également la conductance associée à chaque élément de ce maillage. Plusieurs modules sont donc présents afin d'écrire, de lire et de manipuler ces champs ainsi que les données qu'ils comportent. Il existe un module « Fieldreader » qui permet de lire les données géométriques d'un fichier en mémoire alors qu'un autre module, « FieldWriter », sert à écrire dans un fichier les informations contenues dans un champ. La plupart de données contenues dans SCIRun sont manipulées sous formes de matrices, il est donc primordial que SCIRun possède des modules permettant de les lire « MatrixReader » et d'en écrire « MatrixWriter ». Il existe également un module « SolveMatrix » qui prend en entrée deux matrices et tente d'en déterminer la solution en utilisant différentes techniques, différentes approximations avec des méthodes itératives et de minimisation d'erreurs.

Tel que mentionné précédemment, les champs peuvent être composé du maillage et des données correspondantes; il existe donc un module « ManageFieldData » qui permet l'ajout ou le retrait de données à un champ. En présentant sur le premier port d'entrée un champ et sur le deuxième port une matrice, ce module combine le maillage avec les

données et produira un deuxième champ qui sera présenté sur un de ces ports de sortie. Le module peut également faire l'inverse; i.e. prendre un champ en entrée et le diviser en un champ et en une matrice de données. Ce module est très utile pour un projet tel que celui-ci car il permet de fusionner un maillage géométrique d'un volume défini avec les conductances s'y rapportant. Un autre module utile relié aux champs est le « ChangeFieldDataAt » qui permet de modifier les attributs des données présentées dans le champ d'entrée. On peut, par exemple, définir que le champ d'entrée est un nœud, une surface ou même un volume. Un module similaire permet de modifier le type de données; par exemple, on peut changer une donnée de type « integer » en une de type « float ». Il s'agit du module « ChangeFieldDataType ». Il existe également un module permettant de transformer les données selon les désirs de l'utilisateur : « TransformData ». Ce module permet entre autres, d'ajouter des valeurs dans un champ présenté en entrée.

Le dernier module présenté ici est le « DirectInterpolate » qui prend en entrée un champ primaire contenant la géométrie et les données ainsi qu'un deuxième champ contenant des points d'intérêts. « DirectInterpolate » sert à trouver les valeurs correspondantes à l'intérieur du premier champ à chaque point spécifié dans le deuxième. Ce module est en fait le cœur de notre système; SCIRun doit calculer la tension pour chaque élément du maillage (premier champ présenté), cependant, seulement un certain nombre de ces valeurs nous intéresse; la tension présente autour de chaque axone (deuxième champ).

Une autre classe de modules est présente dans SCIRun concernant l'affichage tridimensionnel. Le module « SampleField » par exemple, permet d'extraire des données de n'importe quel champ en entrée et de les envoyer en sortie sous le format « PointCloudField » (nuage de points). « Streamlines » permet quant à lui de visualiser des vecteurs en interpolant les valeurs entre les vecteurs et en les traçant sous formes de courbes. Le module « GenerateStandardColorMap » attribue des couleurs aux données contenues dans un champ selon une échelle prédéfinie. « RescaleColorMap » modifie

l'échelle prédéfinie du module « GenerateStandarColorMap » à l'échelle des valeurs présentes dans le champ. « ShowField » sert à visualiser la géométrie obtenue à l'aide du maillage contenu dans un champ. Le module utilise les valeurs de conductivité (lorsque présente) pour afficher des couleurs selon les sections du maillage. Finalement, le module « Viewer » permet d'afficher le tout à l'écran.

3.2.2 Modules BioPSE

Tel que mentionné précédemment, SCIRun comporte plusieurs assemblages de blocs destinés à différentes fonctions. L'assemblage BioPSE « Biological Problem Solving Environment » sera utilisé ici afin d'étendre les capacités de SCIRun aux environnements biologiques. Cet assemblage sera utilisé afin de déterminer les valeurs de la tension aux nœuds du maillage en utilisant l'équation de Poisson :

$$\nabla\partial.\nabla\Phi = I_b$$

Équation 8 : Équation de Poisson

Où « ∂ » est la conductivité, « I_b » le courant et « Φ » le potentiel. Afin de résoudre l'analyse des éléments finis, on procède avec l'équation suivante :

$$A\Phi = B$$

Équation 9 : Résolution des éléments finis

Où « A » est la matrice « Stiffness » (approximation de $\nabla\partial.\nabla$), « B » est la matrice discrétisée du maillage contenant les valeurs de courant et finalement, « Φ » est la matrice des potentiels recherchés aux nœuds du maillage. La méthode de résolution utilisée par SCIRun/BioPSE est le « Gradient conjugué conditionné ». Le conditionnement est réalisé avec la méthode de « Jacobi ».

La matrice « A » est obtenue en passant un champ contenant le maillage et les valeurs des données dans le module « SetupFEMatrix ». Ce module permet de bâtir la première

matrice à l'aide d'éléments linéaires réservés aux problèmes contenant des champs bioélectriques (discrétisation de l'équation de Poisson dans le cas d'un volume conducteur). Cependant, les valeurs de conductivité contenues dans le champ sont en fait des indices pointant à des valeurs de conductivité contenues dans une table; avant de pouvoir effectuer la discrétisation il faut donc remplacer les indices par les vraies valeurs de conductivité. L'utilisation du module « ModifyConductivities » permet d'effectuer ce changement. Quant à la matrice « B » contenant les valeurs de courant, elle est construite à l'aide du module « ApplyFEMCurrentSource ». Ce module prend en entrée le champ contenant le maillage et les valeurs de conductivités ainsi qu'un dipôle (anode et cathode). Il produit en sortie la matrice « B » contenant les valeurs discrétisées du courant.

Les principaux modules utilisés dans SCIRun ont été décris dans les dernières sections, cependant, une description détaillée de l'interconnexion entre ces modules sera faite au chapitre 4 lors de la description complète du modèle.

3.3 Logiciel de simulation de la conduction neuronale – Neuron

Le logiciel Neuron peut être installé sur n'importe quel plateforme (Windows, Unix, Linux, MacOS) et est disponible gratuitement sur le site Web des universités Duke et Yale. Neuron est un logiciel très polyvalent et a été développé afin de faciliter la simulation de la conduction neuronale et des réseaux de neurones. Il est utile pour tout modèle présentant des caractéristiques de conduction de type « câble » et il prend en compte plusieurs mécanismes biologiques complexes tel les mécanismes transmembranaires de transport d'ions, l'accumulation d'ions et les seconds messagers. Neuron utilise son propre langage de programmation; le « HOC » (High-Order Calculator) et, il est simple et facile de programmer avec ce langage; le niveau de difficulté est le même niveau que le « Basic » mais la syntaxe et la façon d'appeler les fonctions s'apparente beaucoup plus au langage C.

Neuron inclus notamment les modèles classiques de Hodgkin-Huxley (HH) ainsi que le « Integer&Fire (I&F)», les paramètres peuvent être modifiés à tout moment, même lors d'une simulation. Cet avantage permet de déterminer rapidement l'influence d'un paramètre spécifique sur la conduction neuronale dans son ensemble.

Avec le logiciel, un modèle de neurone basé sur les expérimentations de HH se construit de la manière suivante; premièrement, on crée des sections de câbles auxquelles on associe les paramètres du modèle HH. Par la suite, on ajuste les dimensions (3D) de ces sections et on les connecte toutes ensembles, bout à bout. La géométrie des sections 3D influence le calcul des potentiels au travers des dimensions physiques car elles comportent chacune des éléments intrinsèques tels une résistivité. La création d'un modèle de neurone nécessite en fait 3 structures de base : le corps cellulaire (soma), les dendrites et l'axone (ces structures sont déjà implémentées dans Neuron). À chaque fois que l'on crée une structure dans Neuron on doit lui assigner un mécanisme (HH, PAS, I&E, etc); les paramètres de ces mécanismes seront à la base des calculs du potentiel transmembranaire. Par exemple, si l'on veut créer un neurone possédant une dendrite et un axone on crée ces trois structures, on leur applique chacune un mécanisme et on les connectes ensemble (Figure 3-3).

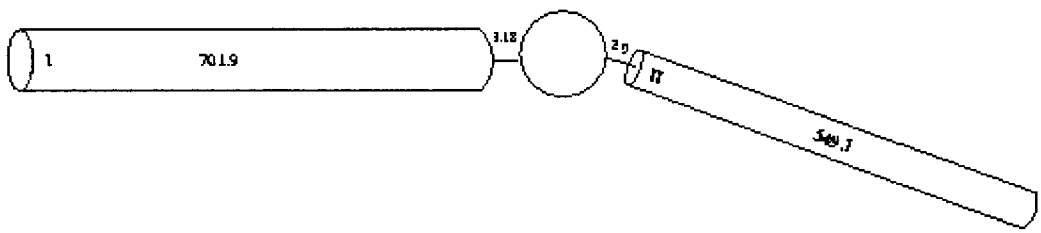


Figure 3-3 : Modèle simplifié d'un neurone avec une dendrite et un axone (Tirée de [10])

Évidemment cette représentation est très simplifiée et on ne peut espérer en tirer des conclusions valables car la valeur des potentiels est calculée par Neuron au milieu des sections. Cependant, on peut, pour chaque structure, définir un nombre de segments afin de subdiviser la structure en éléments plus petits (Figure 3-4). Ainsi, l'effet d'augmenter

le nombre de segments pour une dendrite, par exemple, permet de calculer le potentiel à 3 endroits au lieu qu'à un seul dans la même structure.

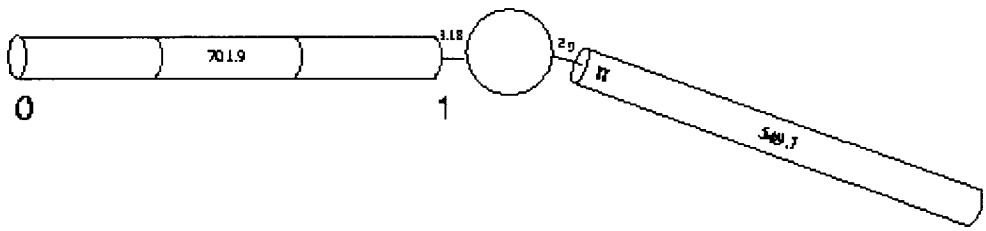


Figure 3-4 : Segementation d'une structure définie dans Neuron (Tiré de [10])

Tel que mentionné précédemment, Neuron permet aussi de construire et de simuler des réseaux de neurones; il est donc possible de créer des structures plus complexes (plus d'une dendrite) et créer plus d'un neurone (Figure 3-5). À l'aide du langage de programmation « HOC » et de l'outil « NMODL » présent dans Neuron, il est possible de modifier ou de créer ses propres modèles que l'on peut insérer dans les différentes structures. Par exemple, reprendre les équations développées par HH et y ajouter le facteur de correction de température.

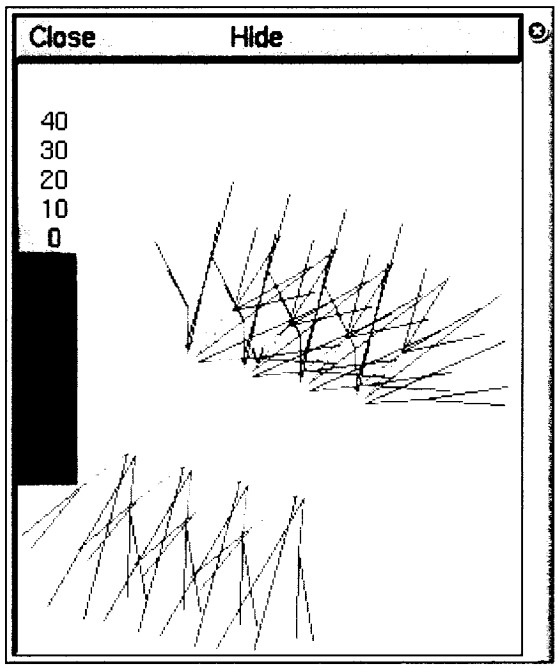


Figure 3-5 : Réseau de neurones du noyau subthalamique chez le rat modélisé à l'aide de Neuron (Tiré de [10])

En ce qui concerne notre modèle, nous n'allons pas modéliser la portion concernant le corps cellulaire (soma) et les dendrites s'y rattachant puisque les PA déclenchés en aval des électrodes sont ceux qui nous intéressent et qui se dirigent vers le muscle. À l'opposé, les PA déclenchés en amont « retournent » vers le soma et le SNC sans aucun effet sur le contrôle de la vessie. Les nerfs sont un regroupement de plusieurs axones émergeant de la colonne vertébrale et se dirigeant vers les organes et/ou muscles; la modélisation se fera donc au niveau du nerf (des axones) principalement.

Tel que mentionné précédemment, la modélisation dans Neuron se fait grâce à des sections de câble interconnectées. Si l'on prend l'exemple d'un axone non myélinisé, on peut le représenter par un segment de câble de longueur définie à l'intérieur duquel on insère les mécanismes nécessaires. La segmentation dans Neuron permet de multiplier et d'augmenter les points stratégiques où le calcul du potentiel est effectué. En effet, la Figure 3-6 montre l'effet d'augmenter le nombre de sections pour un axone non myélinisé avec le mécanisme de conduction ionique proposé par Hodgkin et Huxley (HH). En ajoutant une section, on ajoute également un bloc complet du mécanisme (M2 dans la partie (b) de la Figure 3-6).

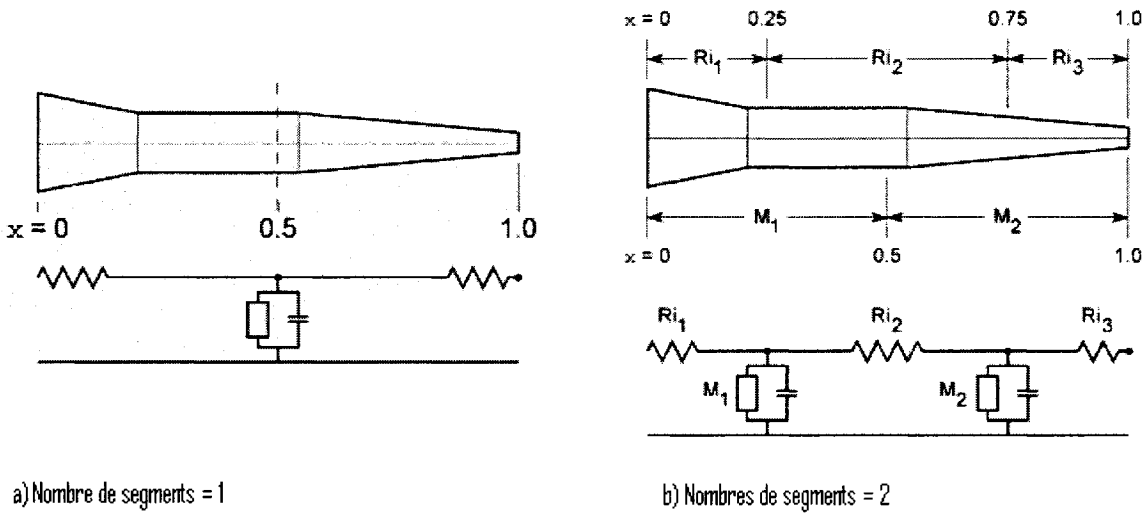


Figure 3-6 : Modélisation d'un axone non myélinisé et l'effet du nombre de segment dans Neuron

Cet ajout double les calculs à effectuer pour la simulation et double également le temps de simulation. Il est primordial de trouver le juste milieu entre précision et temps de simulation.

Passons maintenant à l'exemple d'un axone myélinisé; il est important de prendre en compte les différences majeures au niveau des mécanismes introduits dans un nœud de Ranvier versus ceux de l'espace internodal. En effet, dans le cas d'axone myélinisé, seul les nœuds de Ranvier possèdent des canaux de conductions et jouent un rôle actif au niveau de l'échange des ions avec le milieu extracellulaire. Les espaces internodaux sont considérés comme passifs et ne participent pas à l'échange d'ions entre l'extérieur et l'intérieur. Il est donc conséquent de diviser les nœuds et les espaces internodaux en compartiments indépendants et de les connecter par la suite (Figure 3-7).

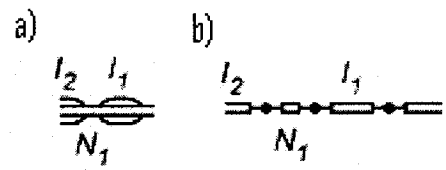


Figure 3-7 : a) axone myélinisé
b) Modélisation de l'axone myélinisé

En effet, en (a) on peut apercevoir la représentation générale d'un axone myélinisé avec un nœud de Ranvier (N_1) et ses espaces internodaux (I_1, I_2). En (b) on voit de quelle façon on reproduit cette morphologie à l'aide de différents compartiments dans Neuron. Tel que mentionné précédemment, on peut subdiviser chaque compartiment en plusieurs sections. L'approche la plus intuitive serait d'insérer les mécanismes de conduction ionique proposés par HH au niveau des sections représentants les nœuds de Ranvier et un mécanisme passif (PAS) au niveau des espaces internodaux. Un autre apport serait de modéliser en plus la couche de myéline qui entoure les espaces internodaux. En réalité, il existe même une certaine fuite d'ions entre cette couche et la membrane de l'axone (l'espace internodal n'est donc pas uniquement passif).

Il existe dans Neuron une fonction prédéfinie appelée « extracellular » qui permet d'ajouter une couche suplémentaire autour de l'axone et de lui attribuer certaines propriétés. Au départ, cette fonction a été créée pour modéliser le milieu extracellulaire entourant les cellules. Cependant, on peut aussi s'en servir pour modéliser une couche supplémentaire de tissus (myéline par exemple) en modifiant conséquemment les bons paramètres. La prochaine section montre un exemple simple de modélisation avec Neuron.

3.3.1 Exemple de simulation

L'exemple présenté ici montre la valeur du potentiel transmembranaire en fonction du temps (Figure 3-8) d'une section représentant le soma. Ce corps cellulaire possède une superficie de 100 μm^2 et une électrode de type « Clamp » est insérée au milieu. On voit que le potentiel de repos se situe approximativement à -70mV et que la dépolarisation engendre un accroissement du potentiel transmembranaire jusqu'à un plafonnement autour de +40 mV. Ensuite, on voit la phase de repolarisation avec retour progressif du potentiel transmembranaire à sa valeur de repos.

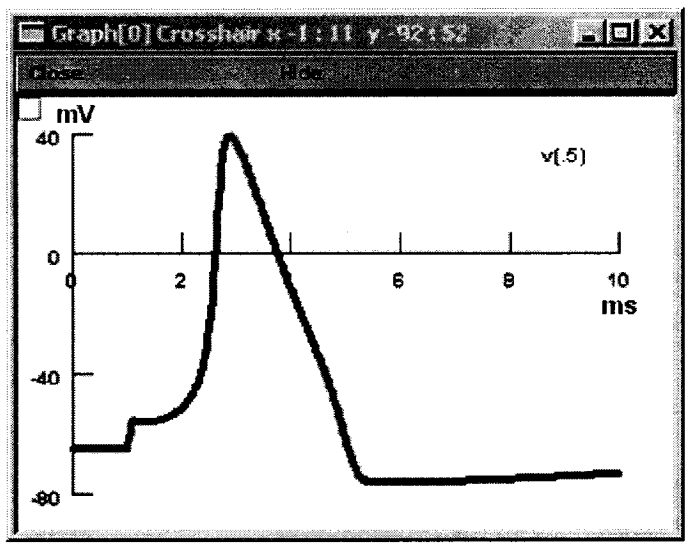


Figure 3-8 : Potentiel transmembranaire d'un soma en fonction du temps

L'interface usager (GUI) de Neuron permet la modification des paramètres de simulation très rapidement. La fenêtre « Parameters » (Figure 3-9) permet de modifier la géométrie du corps cellulaire, les constantes d'activation et de désactivation pour chacun des canaux de conduction d'ions ainsi que les potentiels de repos pour chacun de ces ions. Après chaque modification, on doit réinitialiser la simulation et la redémarrer.

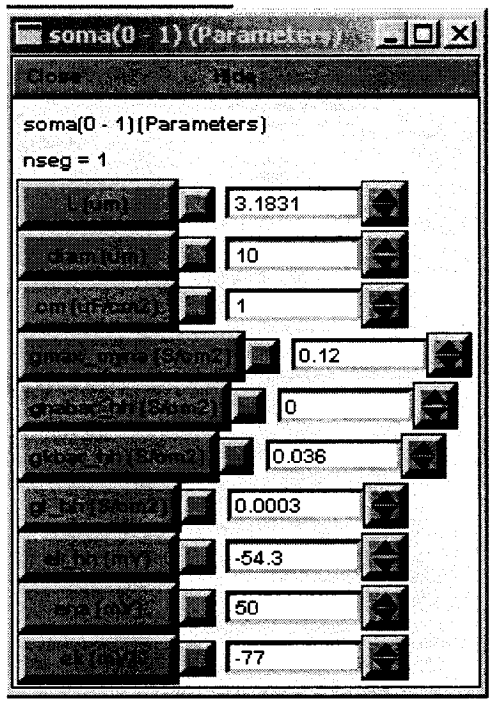


Figure 3-9 : Interface usager pour modifier les différents paramètres morphologiques

La fenêtre « RunControl » permet d'initialiser, de démarrer (Init & Run) et d'arrêter (Stop) les simulations (Figure 3-10). Elle permet également de modifier certains paramètres importants tel que le temps de simulation (Continue til (ms)), le nombre de points à tracer par milliseconde (Points plotted/msec), la valeur du potentiel transmembranaire de départ (Init (mV)), etc. Il est également possible de réaliser une simulation pas à pas en modifiant la valeur d'un paramètre (Continue for (ms)). Un autre paramètre (dt (ms)) fixe le pas de calcul pour les simulations. Le potentiel trans-membranaire est recalculé dans toutes les sections et ce, à chaque intervalle de temps

correspondant à la valeur donné parr ce paramètre (dt (ms)). Plus il est faible, meilleur sera la précision, mais plus élevé sera le temps de simulation.

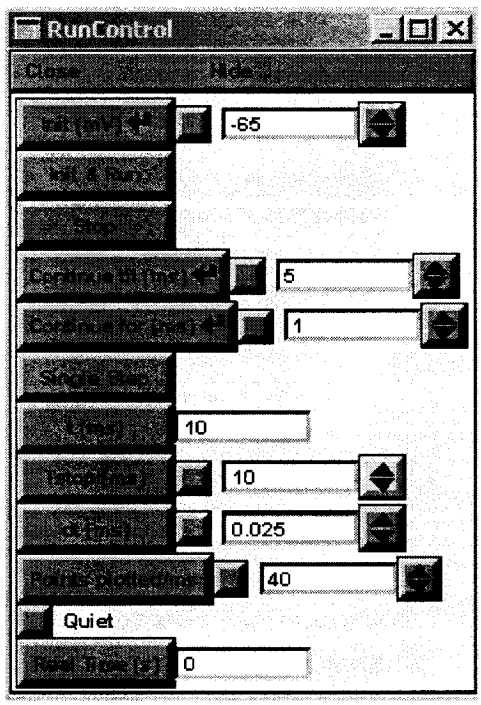


Figure 3-10 : Ajustement des paramètres de simulation

La dernière fenêtre présentée ici est le « PointProcessManager » qui permet de contrôler la nature et le nombre des sites de stimulation. En effet, Neuron possède quelques mécanismes de stimulation (électrodes) déjà implantées dont les plus connus : IClamp (pince de courant) et VClamp (pince de tension). On peut, à l'aide de cette fenêtre, contrôler les différents paramètres de stimulation tel la durée (dur (ms)), l'amplitude (amp (nA)) et le retard (del (ms)) avant que les charges ne soient appliquées.

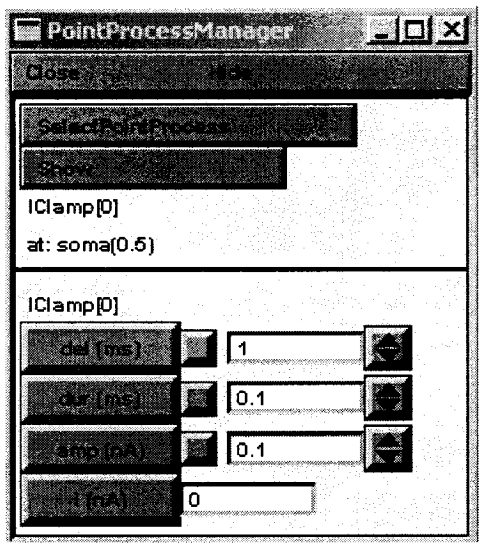


Figure 3-11 : Mode de stimulation

Neuron permet également de contrôler tous ces paramètres à l'aide d'une ligne de commande; on peut écrire dans un fichier toutes les lignes de commandes désirées et « charger » ce fichier lors du démarrage de Neuron. Ceci évite d'avoir à redéfinir les paramètres manuellement à chaque simulation en plus d'en faciliter la gestion lorsqu'un grand nombre de répétitions est requis. En changeant uniquement les variables dans les fichiers, on peut quasiment automatiser les simulations. Une caractéristique importante de Neuron est le langage NMODL qui permet de réaliser pratiquement tout ce dont on a besoin. En effet, à l'aide de NMDOL on peut créer nos propres mécanismes de transfert d'ions, créer une électrode sur mesure, des synapses, etc. C'est un outil très puissant car, dans le domaine de la neuromodélisation, les connaissances sont assez limitées et il est probable que de nouvelles découvertes soient faites; il faut pouvoir adapter et modifier au besoin ce que nous possédons déjà en plus de pouvoir créer de nouvelles fonctions.

3.4 Application permettant de générer le nerf en 3D

Ce dernier logiciel que nous avons utilisé est une application de type « MS-DOS » réalisée en C++ spécialement pour le projet. Le « Peripheral Nerve Builder » a été créé afin d'avoir un outil permettant de générer la structure du nerf. Cette application permet de définir la géométrie des fascicules et de les positionner en 2D (x,y) à l'intérieur du nerf. De plus, elle génère et positionne automatiquement les axones à l'intérieur des fascicules. Elle permet à l'utilisateur de définir plusieurs paramètres tel le diamètre du nerf, l'épaisseur des couches de tissus biologiques, la position des électrodes, la géométrie des électrodes, etc. Tous ces paramètres permettent de générer le maillage et les fichiers d'entrées nécessaires à SCIRun/BioPSE.

3.5 Conclusion

Dans ce chapitre, nous avons abordé les logiciels qui ont servi à l'élaboration de ce modèle et qui permettent d'obtenir des simulations. L'ordre d'utilisation est le suivant : en premier, on définie la morphologie et le maillage à l'aide du « Peripheral Nerve Builder ». Les fichiers générés serviront d'entrée pour SCIRun et pour Neuron. On lance ensuite une simulation avec SCIRun afin d'obtenir les potentiels aux différents points d'intérêts. Ces valeurs sont extraites et servent d'entrés à Neuron. Finalement, on termine avec Neuron afin de simuler la propagation des PA tout au long du nerf. Il est important de mentionner qu l'on peut modifier la fréquence et la largeur d'impulsion de stimulation avec le « Peripheral Nerve Builder » et utiliser les nouveaux fichiers directement avec Neuron. Cependant, si on désire modifier l'amplitude du courant utilisé, il faut revenir dans SCIRun afin d'extraire les nouvelles valeurs de potentiels générés par ce nouveau courant. Le prochain chapitre décrit en entier le modèle réalisé.

Chapitre 4

DESCRIPTION DU MODÈLE DE NERF PRÉSENTÉ

Dans ce chapitre, nous décrirons une vue d'ensemble ainsi que les principes de fonctionnement de notre modèle. Nous allons détailler chacune des parties du modèle en fonction de l'utilisation des différents logiciels.

4.1 Description générale du modèle

Une représentation schématique du modèle et une brève description de chacune des sections est présentée à la Figure 4-1. On peut résumer les différents blocs de la manière suivante :

- « Peripheral Nerve Builder » : Application développé en C++ dans le cadre des présents travaux permettant de construire le nerf et de générer tout les fichiers nécessaires.

- Réseau SCIRun/BioPSE : Réseau construit afin de réaliser l'analyse par éléments finis de la propagation du courant en provenance d'un contact de l'électrode, au travers des couches de tissus constituant le nerf et remontant vers l'autre contact par la prise en charge des fichiers et des matrices créés par le « Peripheral Nerve Builder ». Résolution des équations matricielles et affichage 3D.

- Neuron : Représentation détaillé des mécanismes de transport d'ions transmembranaire ainsi que de la géométrie des axones. Il prend en entrée les valeurs de potentiel trouvées précédemment et résout les différentes équations ioniques en fonction du temps.

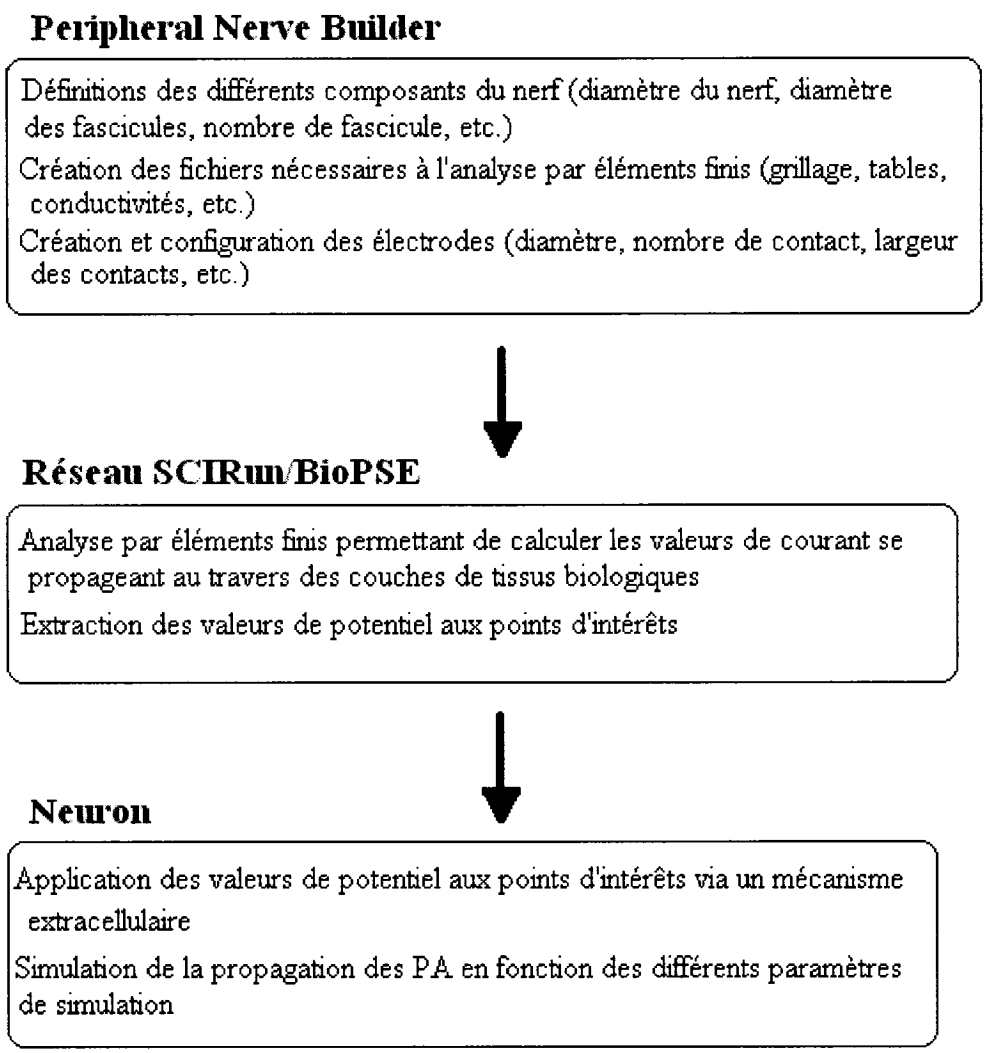


Figure 4-1 : Description schématique du modèle

4.2 Description du « Peripheral Nerve Builder »

La première étape fut de reprendre le modèle d'axone présenté par l'équipe de Grill (voir section 2.2) et de le modifier afin de s'en servir comme base pour notre modèle. Ces changements n'ont pas modifié les équations ni les fondements de leur modèle, mais bien la façon de l'utiliser. L'équipe de Grill ont réalisé leurs simulations avec un seul axone et avec une électrode de type « pince de courant » (Current Clamp). Certaines modifications nous ont permis de transformer leur modèle en un patron (un « template ») qu'on pouvait

réutiliser afin de créer plusieurs instances de cet axone. Par la suite, nous avons travaillé à générer un fascicule tridimensionnel à partir du patron de l'axone. Il fallait également prendre en compte les différentes couches de tissus. Finalement, nous avons remplacé l'électrode « pince de courant » par une électrode extracellulaire beaucoup plus près de la réalité d'une électrode de type « cuff » utilisée dans les présents travaux de recherche par notre équipe Urostim.

Une fois ces modifications réalisées au niveau du code, il nous fallait créer une application permettant de créer et de générer le modèle du nerf en 3D selon les paramètres spécifiés par l'utilisateur. Un programme DOS écrit en C++ a été réalisé; le « Peripheral Nerve Builder » (Figure 4-2). L'application comporte une dizaine d'options présentées à l'utilisateur via une fenêtre de commande à l'écran (le code se retrouve à l'annexe I). Toutes les données sont entrées sur le clavier par l'utilisateur. Les fichiers générés se retrouvent dans le même dossier que l'application elle-même. Une description de chacune des options présentes sera faite dans les paragraphes suivants, les numéros d'options font référence à la Figure 4-2

Les options 1 à 3 permettent la création des fascicules avec la géométrie désirée (rayon et position du centre en x et en y) ou de les effacer, ainsi que de visionner les informations relatives aux fascicules déjà créés. Ces trois options ne génèrent pas de fichiers de sortie, mais plutôt des fichiers temporaires dans lesquels seront prélevées plus tard les informations nécessaires. La première option définie également la courbe de distribution des axones à l'intérieur des fascicules (courbe binomiale [9] définie par défaut). On peut spécifier une nouvelle distribution, utiliser l'ancienne ou utiliser celle prédéfinie. Par contre, si on utilise une distribution différente de celle prédéfinie, il faut modifier le code Neuron puisque les diamètres d'axones présents dans le modèle sont en fonction de cette distribution. De plus, dans le coin supérieur droit de l'interface usager, on retrouve une ligne qui affiche le nombre de fascicules présents jusqu'à maintenant.

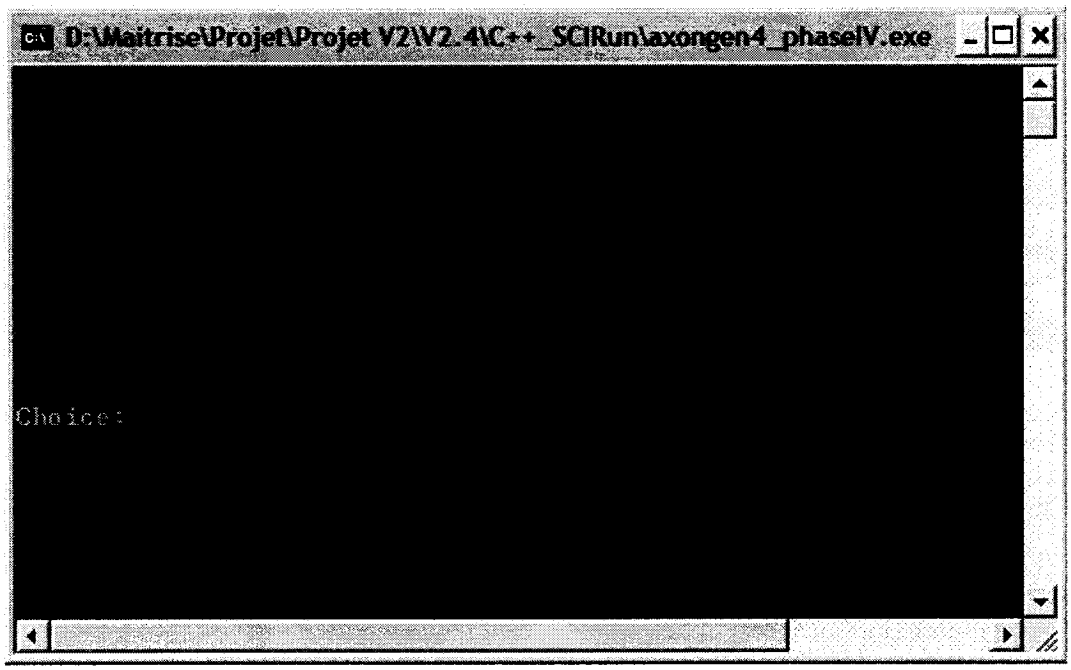


Figure 4-2 : Interface du « Peripheral Nerve Builder »

Les options 4 à 6 permettent de générer les fichiers nécessaires à la résolution des éléments finis; l'option 4 (Create locus file) crée un fichier contenant toutes les informations pour les patrons d'axone et de fascicule. En effet, ce fichier est une matrice de 3 colonnes sur un nombre de lignes variables; la première colonne contient le diamètre de l'axone, la deuxième et la troisième colonne donnent la position (x,y) du centre de chaque axone. De plus, la première ligne du fichier indique le nombre de fascicules présents dans le nerf et la deuxième indique le nombre d'axones présents dans le premier fascicule suivi de la matrice à trois colonnes pour ce premier fascicule. Ensuite, un deuxième chiffre indique le nombre d'axones présents dans le deuxième fascicule, suivi de sa matrice et ainsi de suite.

L'option 5 (Create Statistics file) génère un fichier contenant de l'information sur la géométrie du nerf; les numéros des fascicules, la position de leur centre en x et en y, le nombre d'axones de chaque diamètre, le nombre total d'axones à l'intérieur de chaque

fascicule, l'aire totale occupée par chacun des fascicules ainsi que le ratio de l'aire occupée par les axones versus l'aire totale du fascicule.

L'option 6 (Create Mesh) permet de créer les fichiers contenant les informations du maillage qui serviront d'entrée à SCIRun/BioPSE. Cette option permet de définir le type de maillage (tétraédrique ou hexaédrique), de générer les fichiers comportant tous les points du maillage et également de définir la table des conductivités pour les différents tissus biologiques. On peut soit créer une nouvelle table, soit utiliser celle prédéfinie [28]. Il est important de noter que le réseau réalisé avec SCIRun/BioPSE ne supporte que les maillages tétraédriques pour l'instant.

L'option 7 (Create electrode) permet de définir les paramètres de l'électrode de stimulation; soit son positionnement dans l'axe du nerf, l'espacement entre l'anode et la cathode, la largeur des contacts ainsi que la précision voulue pour l'espacement des points constituants l'électrode. Tel que mentionné précédemment, nous avons modifié l'électrode originelle (pince de courant) pour une matrice de plusieurs points distincts (point source electrode) entourant le nerf à l'extérieur (Figure 4-3). Tous ces paramètres sont définis dans trois fichiers distincts; un pour l'anode, un pour la cathode et un autre pour l'amplitude du courant. Ces trois fichiers serviront de paramètres d'entrées pour le réseau réalisé avec SCIRun/BioPSE.

L'option 8 (Add tripolar reading electrode) n'est pas implémentée dans le réseau SCIRun/BioPSE pour l'instant. Cependant, elle génère trois fichiers semblables à ceux créés par l'option « Create electrode ». Des fichiers contenant les différents paramètres pour chacun des trois contacts; positionnement, espacement entre les contacts, etc. Cependant, aucun fichier contenant les informations pour le courant de stimulation n'est créé puisqu'il s'agit d'une électrode de lecture.

L'option 9 (Add measurement site) permet la création d'un fichier qui sera lu par le réseau réalisé avec SCIRun/BioPSE et qui lui indique à quels endroits il doit prendre ses mesures; i.e. à quels nœuds du maillage les valeurs de potentiels devront être enregistrées. Il contient donc la position tridimensionnelle de chaque axone afin que la résolution des éléments finis nous donne les valeurs de potentiels autour de chacune des axones. On indique à partir de quelle position et jusqu'à laquelle (dans l'axe des z) on veut que les mesures de potentiels soient extraites, on doit également spécifier la précision désirée (valeur de l'incrément pour parcourir l'axe des z).

L'option 10 (Create potentials matrix) sert uniquement une fois que le fichier contenant les valeurs de potentiels nous est retourné du réseau SCIRun/BioPSE. En effet, cette option modifie le fichier de sortie du réseau en y incorporant les données concernant la fréquence de stimulation et le temps de simulation.

La dernière option nous permet de quitter l'application, mais lorsque certaines opérations n'ont pas été effectuées, quelques questions sont posées à l'utilisateur afin de s'assurer que c'est volontaire et qu'il n'a pas oublié d'effectuer ces opérations.

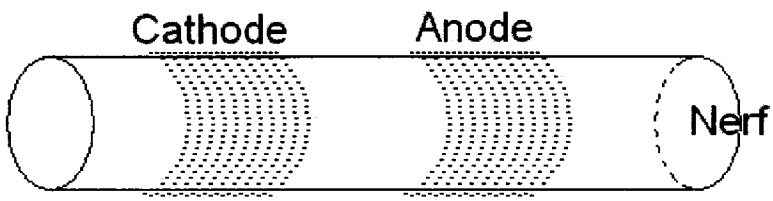


Figure 4-3 : Représentation de l'électrode

En résumé, le « Peripheral Nerve Builder » nous permet de construire le nerf en trois dimensions ainsi que de générer tous les fichiers nécessaires par la suite (pour le réseau SCIRun/BioPSE et pour Neuron). Une fois tous ces fichiers générés, il faut passer au logiciel de résolutions d'éléments finis; SCIRun/BioPSE.

4.1.1 Description du réseau construit avec SCIRun/BioPSE

Afin d'obtenir les valeurs de potentiels autour de chacun des axones, nous avons construit un modèle d'un nerf représenté par un volume conducteur. À l'intérieur de ce volume, nous tenons compte de l'effet des différentes conductivités de chacun des tissus biologiques présents. Tel que mentionné précédemment, la méthode utilisée est la résolution des éléments finis (Finite Element Analysis - FEA). Avec cette méthode, il faut d'abord diviser le volume conducteur en plusieurs sous éléments formant un maillage. Cette opération est réalisée par le « Peripheral Nerve Builder » tel que décrit à la section précédente. Chacun des sous éléments constituant le maillage possède une conductivité propre relative à la portion de tissu représenté par cet élément. Le logiciel utilisé pour la résolution des éléments finis de ce volume conducteur, peut calculer les valeurs de courants et de potentiels à chacun des nœuds du maillage.

Pour ce faire, nous avons construit un réseau à l'aide des blocs prédéfinis de SCIRun/BioPSE. Ce réseau prend en entrée tous les fichiers générés précédemment par le « Peripheral Nerve Builder » et ensuite calcule la propagation du courant au travers des différentes couches de tissus biologiques du nerf. Dans le cas présent, seul les valeurs de potentiel au pourtour de chaque axone sont intéressantes. Le réseau ainsi construit est une cascade de blocs inter reliés qui lisent des fichiers en entrées et qui les transforment. La sortie de chaque bloc devient ainsi l'entrée d'un prochain. Plusieurs des blocs et des connexions servent principalement à afficher le volume conducteur et les lignes de potentiels en trois dimensions à l'écran. Dans les prochains paragraphes, nous allons décrire les connexions majeures afin de clarifier le fonctionnement du réseau.

Premièrement, le fichier contenant le maillage tétraédrique est lu par un module « FieldReader » (Figure 4-4). Le maillage est transformé en un champ à l'aide de deux autres fichiers; un fichier « .pts » contenant les coordonnées de tous les nœuds du maillage ainsi qu'un fichier « .tet » contenant les indices correspondant à chacun des nœuds des tétraèdres du volume conducteur. Ensuite, la matrice contenant les indices des

différentes valeurs de conductivités est lue via le bloc « MatrixReader » (Figure 4-4). La sortie des ces deux modules entre dans le bloc « ManageFieldData » (Figure 4-4) qui combine le champ contenant l'information géométrique avec la matrice des indices de conductivités en un champ contenant à la fois des valeurs géométriques et des données. Le module « ChangeFieldDataType » (Figure 4-4) qui prend en entrée ce champ combiné, remplace les indices de conductivités par une valeur « entière » afin que les indices puissent être interprétés plus tard comme un indice, mais non pas comme la valeur même de conductivité. Un deuxième bloc « MatrixReader » (Figure 4-4) prend en entrée la matrice contenant les conductivités des différentes couches de tissus et l'envoi au bloc « ModifyConductivities » qui prend également en entrée le champ combiné décrit précédemment. Ce module remplace l'indice des conductivités présent dans le champ combiné par la vraie valeur de conductivité et rend disponible à sa sortie ce nouveau champ combiné. Celui-ci est alors distribué à plusieurs autres blocs, mais principalement au bloc « SetupFEMatrix » qui construira la matrice « stiffness A ». Le champ est également envoyé au bloc « ShowField » afin de générer une représentation tridimensionnel affichable à l'écran.

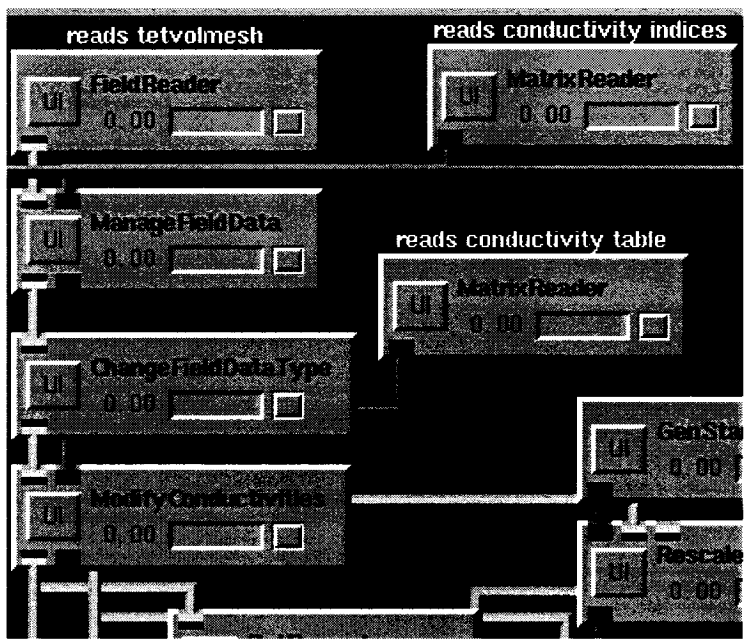


Figure 4-4 : Blocs et connexions qui permet de lire les fichiers contenant le maillage et les valeurs de conductivités.

Deuxièmement, les paramètres de l'électrode de stimulation sont insérés, en premier lieu, le fichier contenant les informations de l'anode est lu par le bloc « FieldReader » (Figure 4-5). Même procédé pour le fichier contenant les informations de la cathode (Figure 4-6). L'information concernant l'amplitude du courant de stimulation est contenue dans une matrice qui est lue par le bloc « MatrixReader » (Figure 4-6). Ensuite, deux blocs « ManageFieldData » (Figure 4-5 et 4–6) sont utilisés pour combiner en un seul champ la géométrie de l'électrode et l'intensité du courant. Ce champ est ensuite dirigé vers 2 blocs « ApplyFEMCurrentSource » (Figure 4-5 et 4–6) interdépendants qui créeront la matrice « Right Hand Side » (ou la matrice « B »).

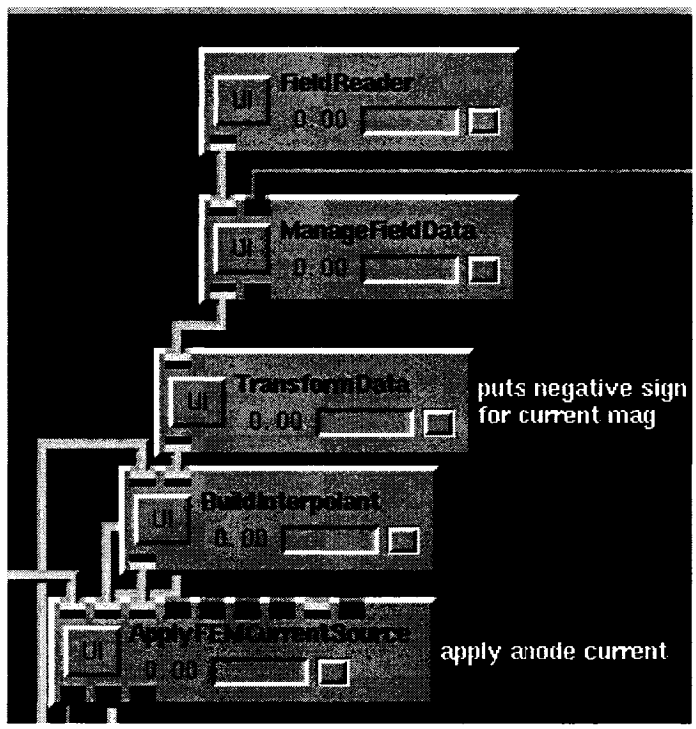


Figure 4-5 : Blocs et connexions nécessaires pour l'anode

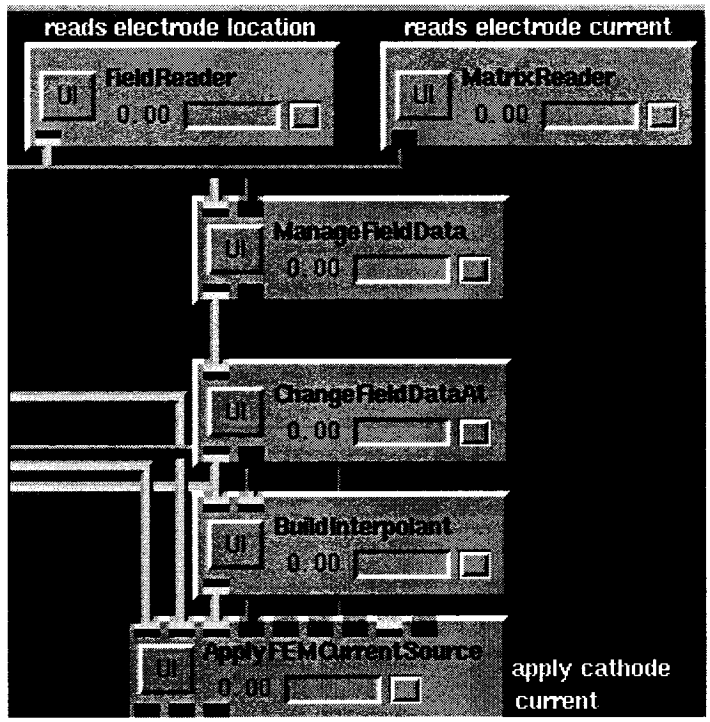


Figure 4-6 : Blocs et connexions nécessaires pour la cathode et l'amplitude du courant

Finalement, une fois les deux matrices A et B créées, elles sont dirigées vers l'entrée d'un bloc « SolveMatrix » (Figure 4-7) qui résout l'équation Ax = B afin d'obtenir les valeurs de potentiels (x) aux nœuds du maillage et place ces données dans une matrice colonne.

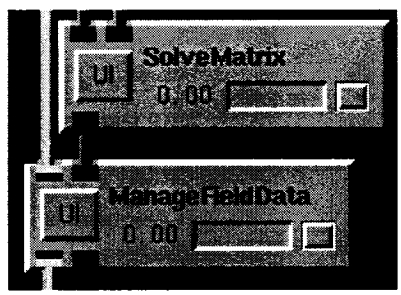


Figure 4-7 : Blocs servant à solutionner et à obtenir les valeurs de potentiels

Afin d'obtenir les valeurs de potentiels autour des axones, les points géométriques d'intérêts sont lus par le bloc « FieldReader » (Figure 4-8) et sont envoyés avec le maillage combiné et la solution vers le bloc « DirectInterpolate » (Figure 4-9). Celui-ci

donne en sortie un champ contenant les informations géométriques ainsi que les valeurs de potentiels interpolées aux points d'intérêts. Ensuite, les valeurs de potentiels sont séparées des informations géométriques à l'aide du bloc « ManageFieldData » et inscrites dans une matrice colonne à l'aide du bloc « MatrixWritter ».

Le fichier obtenu en sortie (potentials.mat) doit être converti en un fichier « .txt » par une fonction prédéfinie (ColumnMatrixToText) dans SCIRun appelée directement à partir de la ligne de commande. On obtient alors le fichier « potentials.txt » contenant les valeurs de potentiels autour de chacun des axones du nerf traité. À l'aide de l'option 10 du « Peripheral Nerve Builder », on modifie à nouveau ce fichier afin de lui incorporer la fréquence de stimulation et le temps de simulation. Ce fichier sera ensuite utilisé par Neuron afin de simuler la propagation des PA.

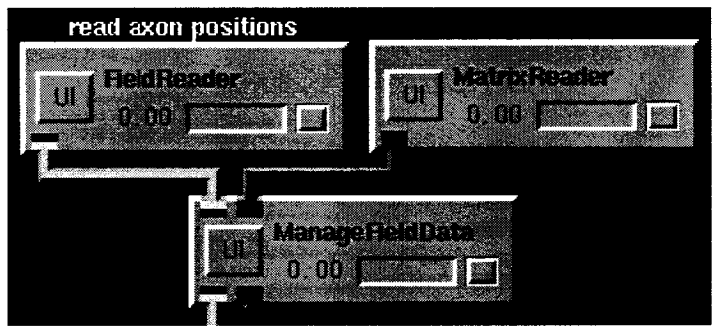


Figure 4-8 : Blocs qui permet la lecture des points d'intérêts

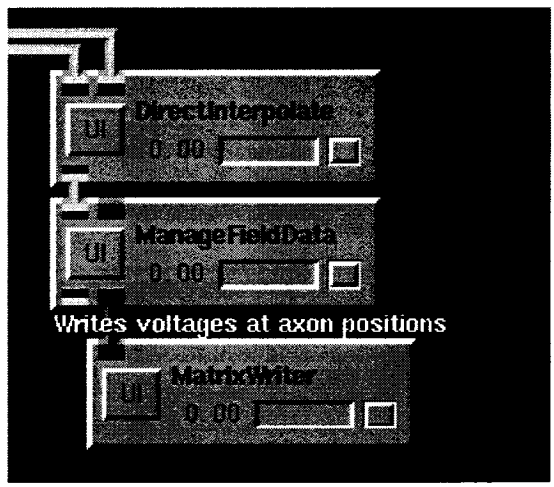


Figure 4-9 : Interpolation des valeurs de potentiels et écriture des valeurs dans un fichier

4.1.2 Code et simulations avec Neuron

Tel que mentionné précédemment, le modèle d'une électrode de type « pince de courant » qui a été adopté dans les travaux de McIntyre et al. a été modifié afin d'obtenir une configuration d'électrode plus proche de celle utilisée dans nos travaux expérimentaux. Auparavant, les valeurs de potentiels extraites par SCIRun/BioPSE étaient directement injectées via la « pince de courant » au centre de l'axone, cependant, les valeurs maintenant extraites sont situées autour des axones et non pas au centre de celles-ci. Cette modification touche principalement le code développé pour Neuron. Premièrement, il faut connaître la position en z de chacune des sections de chaque axone. Pour ce faire, nous avons implémenté une boucle qui parcourre tous les nœuds de Ranvier et enregistre leurs positions respectives. Cette boucle est également présente pour les autres sections (endroit où s'attache la myéline à l'axone (MYSA), la portion conique de la myéline (FLUT) ainsi que l'espace internodal (STIN)).

Ensuite, nous avons ajouté une deuxième couche extracellulaire autour de l'axone avec la fonction « insert extracellular » (Figure 4-10). Les composantes « xraxial », « xc » et « xg » de la première couche sont utilisées afin de modéliser les propriétés électriques de la couche de myéline. En choisissant « xg[1] » et « xraxial[1] » très grands et « xc[1] » très petit de la deuxième couche, on peut utiliser le « e_extracellular » comme source de potentiel à l'extérieur de l'axone. Ainsi, il ne reste qu'à assigner les valeurs de potentiels extraites de SCIRun/BioPSE à cette source de potentiel extracellulaire afin de simuler l'effet du potentiel obtenu par une électrode extracellulaire.

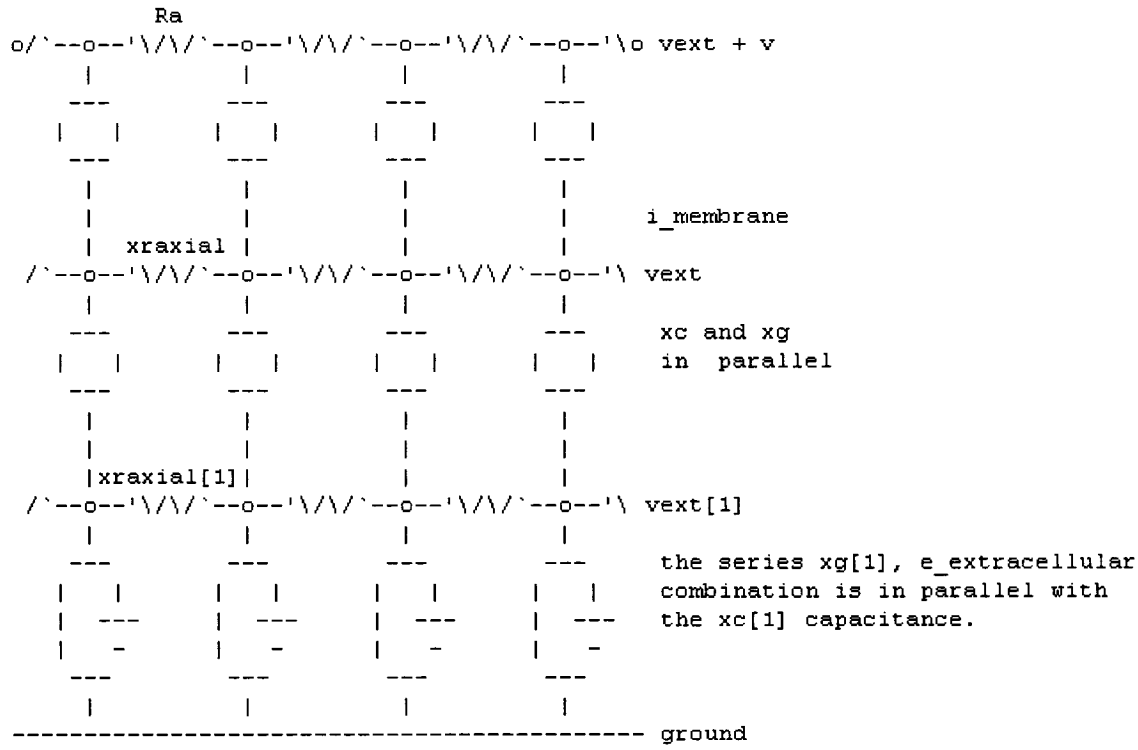


Figure 4-10 : Représentation électrique de la fonction « extracellular » (tiré de [10])

Afin de démarrer une simulation avec Neuron, il faut tout d'abord avoir les fichiers suivants; init.hoc, driver.hoc, mye_axon.hoc, AXNODE.mod, fascicle.hoc, start.ses (annexe III) ainsi que les deux fichiers (locus.txt et potentialsmatrix.txt) provenant du « Peripheral NerveBuilder ». Le fichier « init.hoc » sert uniquement à automatiser l'ouverture des fichiers et le démarrage d'une simulation. Le fichier « driver.hoc » est celui qui place en mémoire le patron de l'axone myélinisé ainsi que le patron du fascicule mais également celui qui lit les données contenues dans le fichier « locus.txt ». Il utilise ces données afin de générer le modèle du nerf en trois dimensions. C'est également lors de la lecture de « driver.hoc » que les valeurs de potentiels contenues dans le fichier « potentialsmatrix.txt » sont associées aux bons axones et attribuées aux sections correspondantes (NODE, FLUT, MYSA ou STIN). Le fichier « AXNODE.mod » contient les équations de courants ioniques pour les nœuds de Ranvier développées à l'origine par McIntyre et al.. Il est appelé à chaque fois que le « mécanisme » est inséré

dans un nœud de Ranvier. Il contient également les différentes valeurs présentées dans les tableaux précédemment ainsi que les équations du facteur de correction de température.

4.1.3 Interface usager lors des simulations

Le fichier « start.ses » définit les principaux paramètres de l'interface graphique de Neuron pour permettre à l'utilisateur d'obtenir différents graphiques; un premier trace l'amplitude des PA en fonction du temps, un autre permet de visualiser la géométrie du nerf, etc. Une fois tous ces éléments présents, une simulation est démarrée automatiquement et les résultats s'affichent au fur et à mesure. Ainsi à la Figure 4-11 on peut voir les différentes fenêtres de Neuron; dans la colonne de gauche et dans l'ordre; menu principal de Neuron, juste en dessous c'est la fenêtre permettant d'imprimer et de gérer l'espace de travail, la troisième montre une coupe transversale du nerf et la dernière est la ligne de commande. La colonne du milieu présente 4 fenêtres affichant les PA qui se propagent, la première présente les PA en fonction de leur position en z et les trois autres en fonctions du temps. Chaque couleur représente un diamètre d'axone différent.

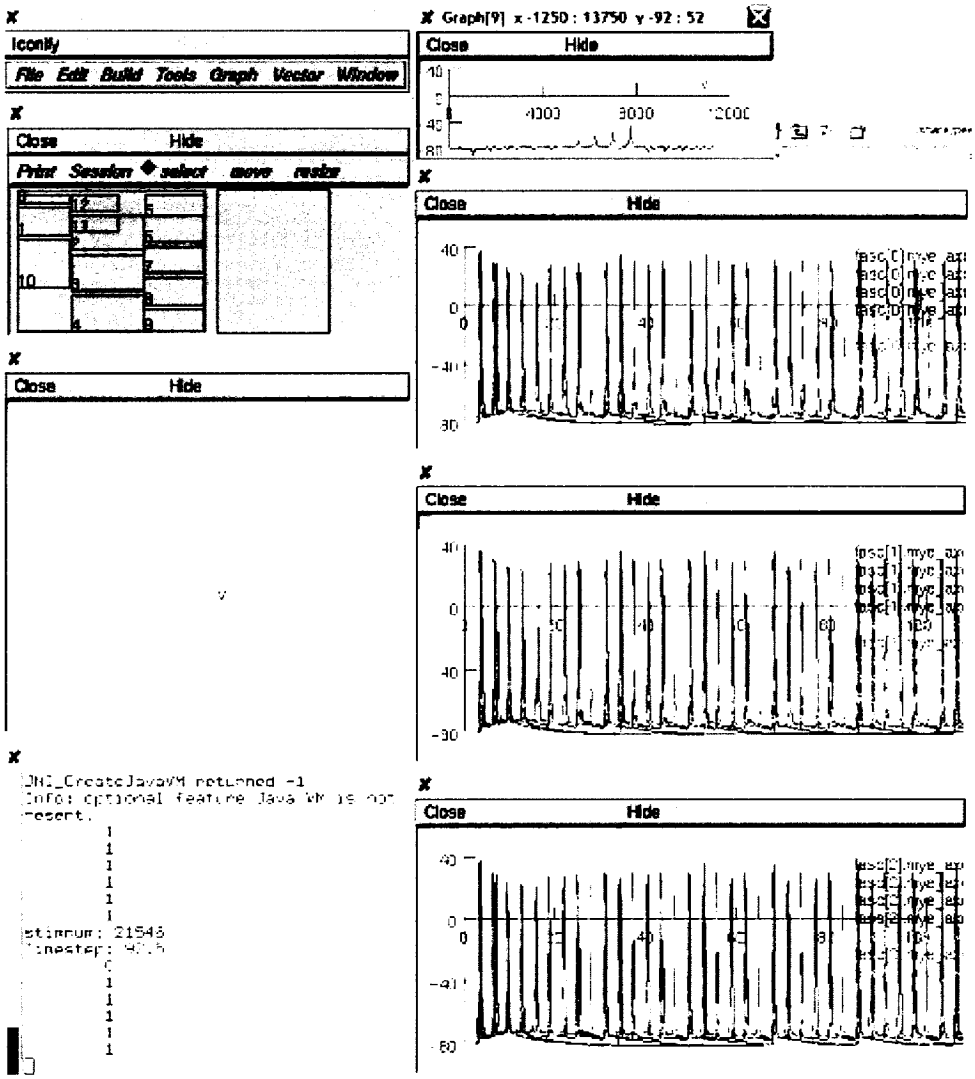


Figure 4-11 : Interface graphique de Neuron

La fenêtre « Shape plot » permet de visualiser la configuration du nerf (Figure 4-12). Dans l'exemple présenté, le nerf possède 8 fascicules de différents diamètres et chacun des points le constituant représente un axone. On peut également noter que les différentes couches de tissus ne sont pas représentées dans Neuron; le modèle développé ici ne tient pas compte de l'interaction entre les axones elles-mêmes lors du passage des PA. Cette décision a été prise suite à une discussion avec un des concepteurs de Neuron (Ted Carnevale) qui a expliqué que ce type d'interaction est très difficile à implémenter et que de par son expérience, cette interaction ne jouait pas un rôle prépondérant dans la

propagation des PA. La complexité requise ne justifiait donc pas l'implémentation de cette interaction. Par contre, avec la résolution des éléments finis via le réseau construit on tient compte des différentes couches de tissus lorsque le courant provenant des électrodes les traverses.

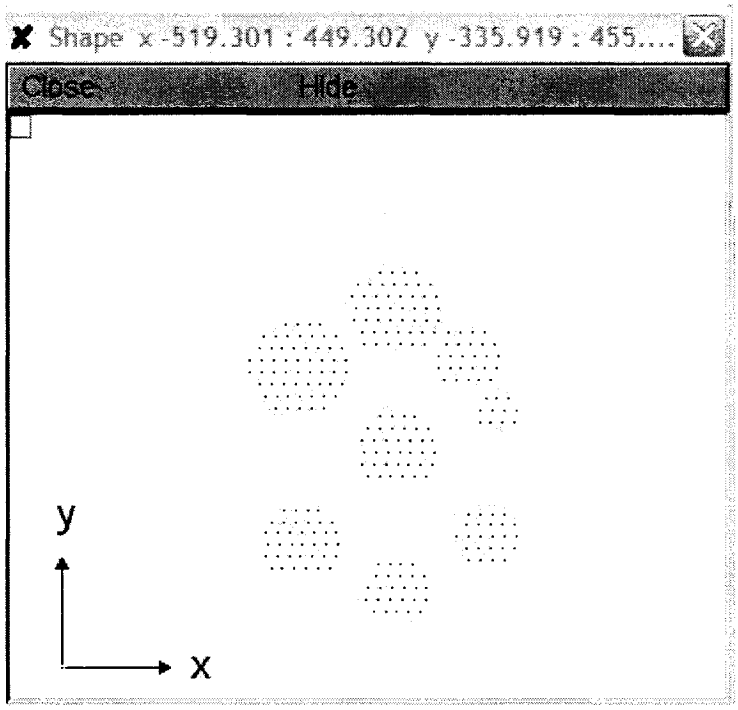


Figure 4-12 : Représentation du nerf dans Neuron

Un graphique très intéressant est celui de la valeur du potentiel transmembranaire en fonction de l'axe des z. On peut, par exemple, observer l'effet du courant provenant de l'électrode sur la variation du potentiel transmembranaire directement au niveau des différentes sections (NODE, MYSA, FLUT ou STIN) composant l'axone (Figure 4-13). Au fur et à mesure que la simulation s'effectue on peut observer le « déplacement » des PA tout au long du nerf (Figure 4-14). Ensuite, les graphiques les plus intéressants sont ceux montrant la valeur du potentiel transmembranaire en fonction du temps pour différents axones. Ce ttype de graphique permet de voir le comportement du nerf via le comportement de ses axones en observant la valeur du potentiel transmembranaire d'axones de diamètres différents situés à l'intérieur des fascicules.

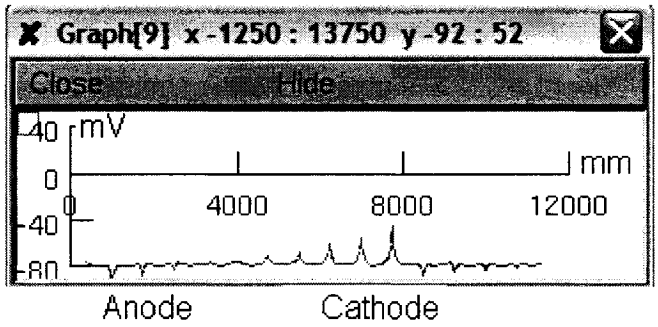


Figure 4-13 : Effet de l'électrode sur le potentiel transmembranaire en fonction de l'axe z.

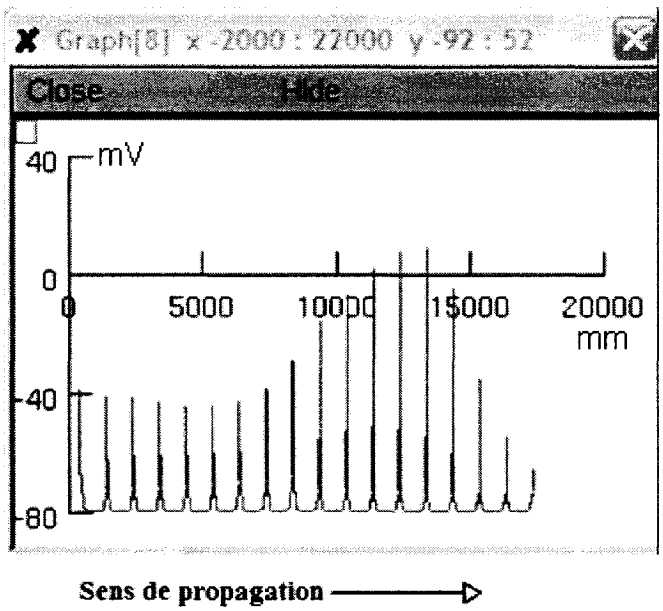


Figure 4-14 : Valeur du potentiel transmembranaire lors du passage d'un PA en fonction de l'axe z

Concernant la présentation des résultats, le point d'observation choisi est l'extrémité de chaque axone (donc l'extrémité du nerf en z) afin d'observer uniquement les PA qui se propagent jusqu'au bout. Le potentiel d'action global (Compound Action Potential - CAP) observé normalement à l'aide d'une électrode de lecture est tout simplement une superposition linéaire des PA dispersés dans le temps. Ainsi, en superposant tous les PA enregistrés, on retrouverait une forme de signal similaire à celle enregistrée lors d'un électroneurogramme (ENG).

4.2 Conclusion

Dans ce quatrième chapitre, nous avons abordé le fonctionnement du modèle présenté en détaillant chacune des étapes nécessaires afin d'obtenir les résultats de simulation prévus. Nous avons également détaillé certains aspects du modèle tel le fonctionnement des électrodes et la manière de présenter les résultats. Il est à noter qu'un guide de l'utilisateur avec un exemple complet de simulation est présenté à l'annexe IV.

Le dernier chapitre présente les résultats de simulations obtenus dans le cadre de ce projet. Premièrement, une validation du modèle sera présentée, suivie des principaux résultats et d'un retour sur la problématique posée au départ.

Chapitre 5

RÉSULTATS DE MODÉLISATION ET DE SIMULATION

Afin de valider les résultats obtenus par notre modèle, nous avons réalisé 3 simulations différentes afin d'en comparer les résultats à 3 résultats expérimentaux. Il s'agit de la même comparaison que l'équipe de McIntyre à présenté dans leur article [16], mais cette fois-ci, avec un nerf complet (comprenant plusieurs fascicules, les différents tissus, etc.) plutôt qu'avec un axone seul. Cette validation occupe la majeure partie de ce chapitre. Ensuite, nous présentons les résultats obtenus lors des simulations effectuées dans le cadre de notre projet d'amélioration de la stimulation sélective.

5.1 Validation du modèle proposé

Avant d'effectuer des simulations dans le but d'obtenir des résultats concernant l'amélioration de la stimulation sélective, il fallait valider notre modèle. Pour ce faire, McIntyre et al., [16] lors de la présentation de leur modèle, avaient effectué plusieurs simulations et avaient comparé les courbes obtenues à des résultats expérimentaux provenant de différents travaux. Afin de valider notre modèle représentant le nerf complet, nous avons effectué les mêmes simulations et nous avons comparé les résultats aux mêmes données expérimentales. Le comportement de notre modèle est très similaire aux comportements obtenus lors des essais expérimentaux.

La Figure 5-1 présente les résultats obtenus en simulation comparativement aux résultats expérimentaux, la première validation a été réalisée à l'aide d'enregistrements effectués sur des rats par Bowe et al. [3]. Lorsque deux impulsions sous-seuil consécutives sont appliquées, la deuxième déclenche un PA (Figure 5-1 A et B). Cette activité « supranormale » est causée principalement par un phénomène passif appelé « dépolarisation post-potentiel », mais également de l'activation des canaux de sodium

persistent proposé par McIntyre et al. [16]. On peut observer que le phénomène dépend du diamètre des fibres (différentes courbes présenté en (B) à la Figure 5-1).

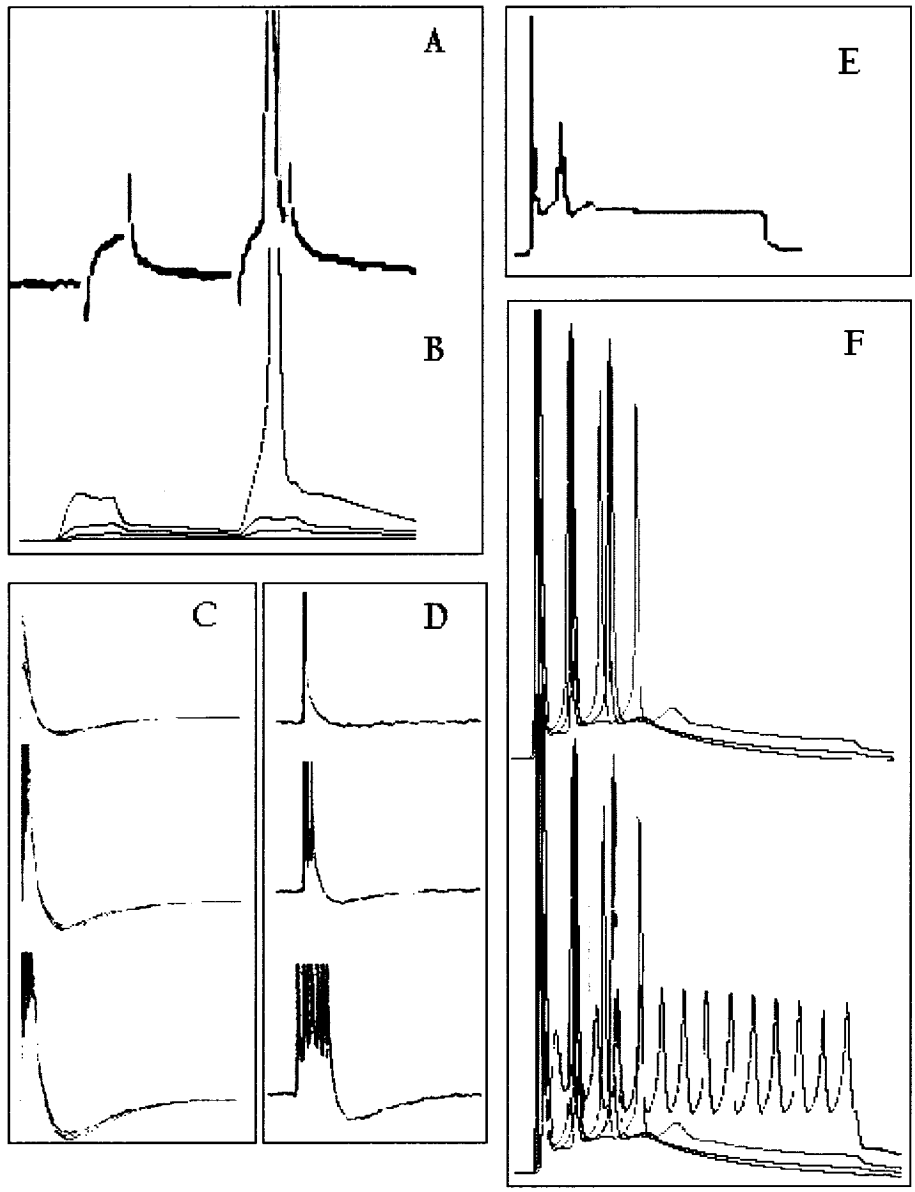


Figure 5-1 : Validation du modèle proposé :
-Deux impulsions sous seuils consécutives séparées par 10 ms; (A) Données expérimentales [3], (B) Données de notre modèle.
-L'amplitude et la durée de la phase AHP augmentent avec le nombre de stimuli; (C) Données de notre modèle, (D) Données expérimentales [2].
-Accommodation fréquentielle; (E) Données expérimentales [2], (F) Données de notre modèle.

Une seconde validation a été réalisée à l'aide de résultats provenant des expériences de Baker et al. [2]. Lorsqu'on augmente le nombre d'impulsions ayant une amplitude supérieure au seuil de déclenchement à une fréquence de 200 Hz, l'amplitude et la durée de la phase d'hyperpolarisation augmentent (Figure 5-1 C et D). On en conclut que l'amplitude et la durée de la phase d'hyperpolarisation qui suit le PA dépendent des impulsions. La réponse de notre modèle est très similaire à celle enregistrée lors des expériences. Les amplitudes sont tronquées sur l'image pour des fins de clarté.

La troisième et dernière validation porte sur l'accommodation fréquentielle; ce phénomène est très souvent rencontré dans les techniques utilisées à des fins de stimulation électrique fonctionnelle. Notre modèle présente une accommodation fréquentielle après 1 à 5 PA. Ces résultats concordent avec les expériences effectuées sur les rats par Baker et al. [2] (Figure 5-1 E et F). Il est important de noter que ce phénomène est dépendant du diamètre des axones; Figure 5-1 (F) chaque courbe représente un diamètre différent.

5.2 Rappel de la problématique

La problématique du départ consistait à prévenir la contraction simultanée du sphincter et du détrusor en utilisant une stimulation neuronale dite sélective. La technique de stimulation sélective introduite par notre équipe Polystim permet d'engendrer une contraction du détrusor tout en inhibant celle du sphincter. Cette stimulation sélective est construite à l'aide de deux trains d'impulsions de courant bipolaires superposés; un train dit « basse fréquence » avec « haute amplitude » permettant de stimuler la vessie et un autre « haute fréquence » avec « basse amplitude » permettant le relâchement du sphincter. La basse fréquence utilisée lors de nos expériences in vivo est de 30 Hz, quant à la haute fréquence, nous utilisons un signal de 600 Hz. Rappelons que le signal haute fréquence permet de bloquer la propagation des PA se dirigeant vers le sphincter externe, il agit, en quelque sorte, de la même manière qu'un influx inhibiteur. Les résultats

obtenus à l'aide de notre modèle présentent la faisabilité d'effectuer une « sélection » au niveau des axones en variant la fréquence du stimulus, mais en incluant un deuxième paramètre qu'est la largeur des impulsions utilisées.

5.3 Résultats du modèle

Notre modèle supporte la théorie énoncée précédemment, mais révèle l'importance de la largeur des impulsions utilisées. Les résultats démontrent que la sélectivité à l'aide d'un blocage haute fréquence dépend principalement du temps de recouvrement des axones suite à une dépolarisation de la membrane. Ce temps de recouvrement varie en fonction du diamètre des axones. Cependant, en utilisant différentes largeurs d'impulsions, on affecte la membrane au niveau des échanges ioniques; par exemple, en utilisant une largeur d'impulsion plus grande, on maintient un déséquilibre au niveau du flux des ions après la phase de repolarisation et on empêche ainsi la membrane de revenir à son état de repos. Pour une même fréquence, plus la largeur d'impulsion est grande, plus la période de temps séparant deux impulsions sera courte. Le temps nécessaire aux ions pour ramener l'équilibre au niveau des flux n'est pas suffisant et empêche en quelque sorte la membrane de retourner dans son état de repos. Ce phénomène est connu sous le nom d'accommodation fréquentielle « spike frequency accomodation ». Un phénomène similaire se produit également lorsqu'un stimulus est appliqué de façon constante pendant une période de temps suffisament longue; l'axone « s'habitue » à la présence du stimulus et cesse de déclencher des PA. On arrive à obtenir cette inhibition en appliquant un stimulus avec une fréquence élevée.

Tous les résultats présentés dans ce chapitre ont été obtenus avec un modèle de nerf possédant les caractéristiques suivantes :

Tableau 5-1 : Caractéristiques du modèle utilisé pour les simulations

Variables	Valeurs
Diamètre du nerf	600 µm
Longueur du nerf	25 mm
Nombre de fascicules	8
Diamètres des fascicules / Nombre d'axones par fascicule	
Fascicule 1	110 µm / 37
Fascicule 2	120 µm / 48
Fascicule 3	90 µm / 26
Fascicule 4	80 µm / 24
Fascicule 5	60 µm / 11
Fascicule 6	90 µm / 26
Fascicule 7	130 µm / 57
Fascicule 8	110 µm / 37
Diamètre des axones (µm)	5,7 – 7,3 – 8,7 – 10 11,5 – 12,8 – 14 15 – 16
Diamètre de l'électrode	640 µm
Espacement Anode-Cathode	3 mm (centre à centre)
Largeur des contacts	1 mm
Forme du signal	Rectangulaire
Amplitude de stimulation	1 mA
Largeurs d'impulsion	100, 175 et 500 µsec
Épaisseur du Périneurium	10 µm
Épaisseur de la couche de gras entourant le nerf	20 µm

À la Figure 5-2 on voit clairement que les axones de diamètres plus élevés (7,3 à 16 µm) continuent de produire des PA alors que les plus petits axones (diamètre de 5,7 µm) n'en génèrent plus (courbe située au bas de l'image). En simulant le comportement des axones avec plusieurs combinaisons de paramètres (fréquence et largeur d'impulsion) on devrait être en mesure de tracer une « Courbe de sélectivité » qui permettrait de déterminer les paramètres nécessaires afin d'inhiber les PAs générés par les axones en fonction de leur diamètre. On imagine un graphique de référence permettant de déterminer les paramètres nécessaires pour activer sélectivement les axones.

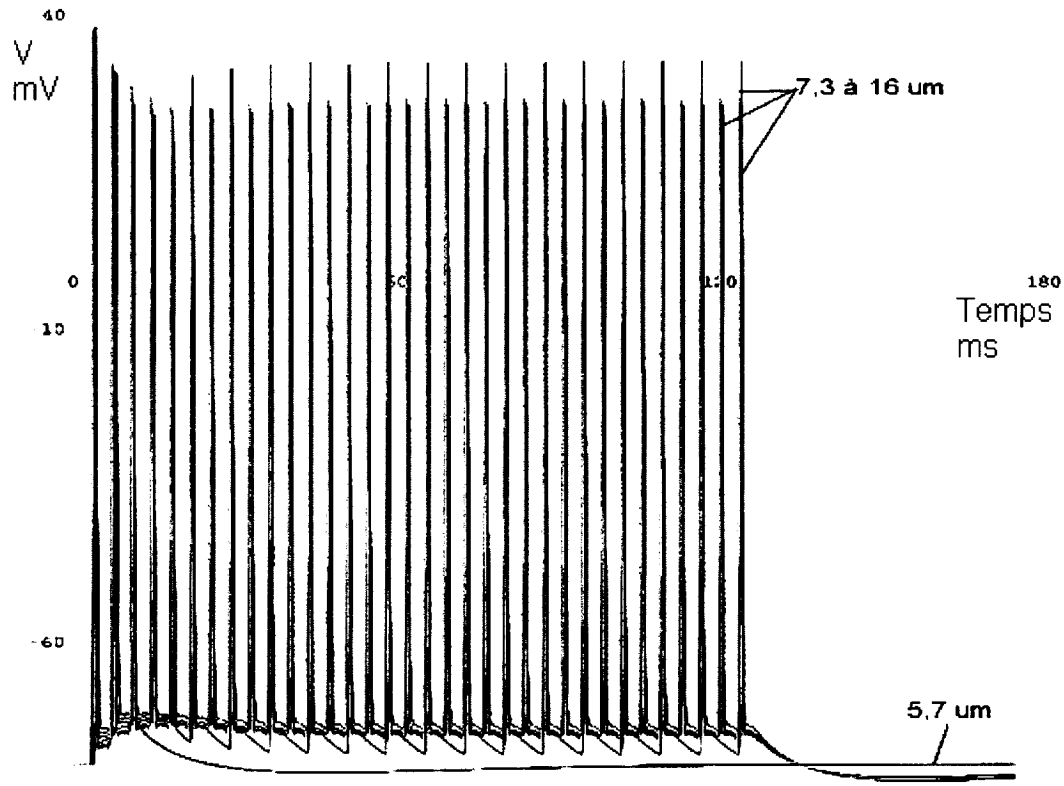


Figure 5-2 : PA en fonction des différents diamètres d'axones (fréquence de 275Hz, largeur d'impulsion de 500µs et amplitude de stimulation de 1mA)

Avec une largeur d'impulsion de 500 µs, au fur et à mesure que l'on augmente la fréquence de stimulation, les axones possédant les diamètres les plus petits jusqu'aux plus grands, arrêtent de décharger des PA (Figure 5-3). Ainsi, avec une combinaison de

cette largeur d'impulsion et d'une fréquence de stimulation de 500 Hz, les axones possédant un diamètre inférieur ou égal à 10 um, sont carrément inhibés (Figure 5-3).

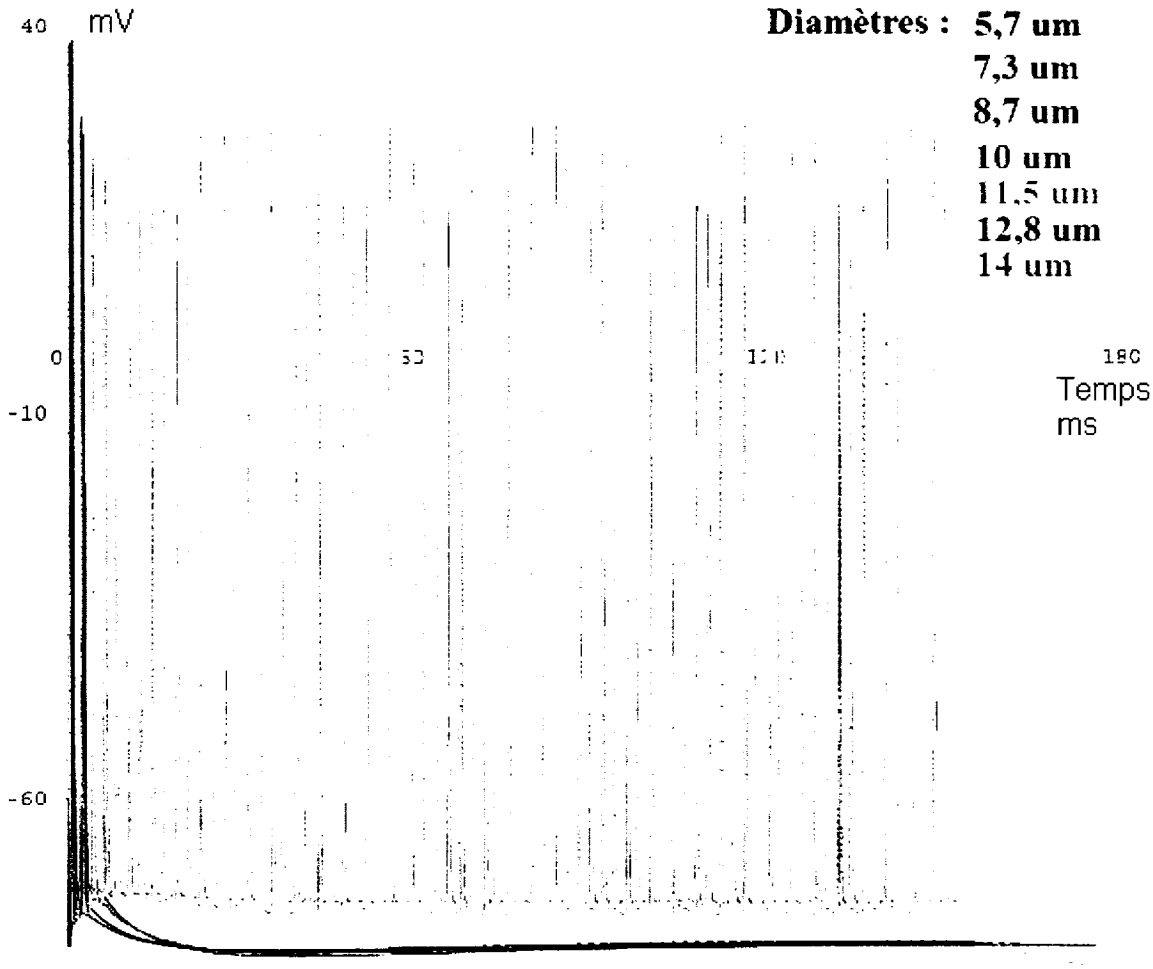


Figure 5-3 : PA en fonction des différents diamètres (fréquence de 500 Hz, largeur d'impulsion de 500 µs et amplitude de stimulation 1mA)

La Figure 5-4 présente la distribution des axones, en fonction de leur diamètre et ce pour chacun des fascicules. Rappelons qu'il s'agit d'une distribution binomiale avec un premier maximum situé entre 7,3 et 8,7 µm et un deuxième situé entre 11,5 et 12,8 µm. On peut remarquer que certains fascicules ne possèdent pas d'axones de chaque diamètre, ce qui concorde parfaitement avec la littérature.

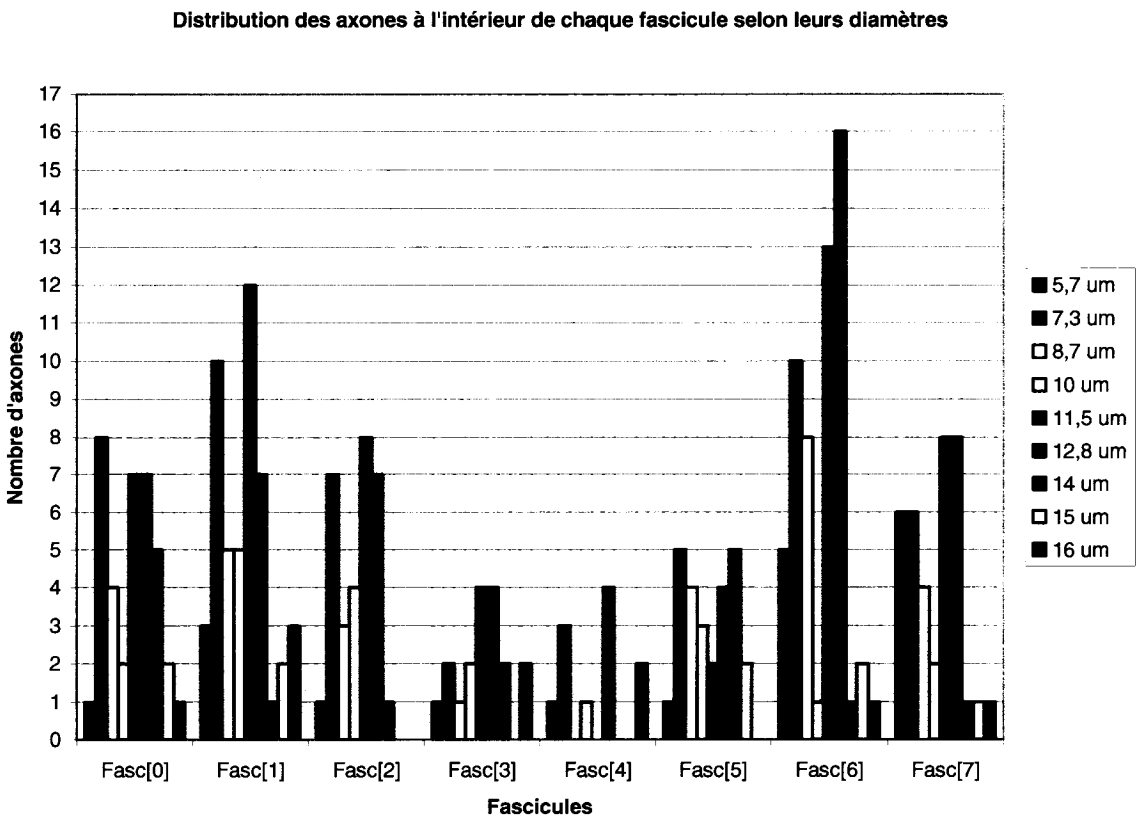


Figure 5-4 : Distribution des axones à l'intérieur des fascicules

Les résultats de simulation obtenus ont été sauvegardés et analysés afin de pouvoir tracer les « Courbes de sélectivité » des axones en fonction de leurs diamètres. La Figure 5-5 présente un graphique comprenant 3 courbes; chacune d'elle délimite la fréquence minimale nécessaire pour inhiber le déclenchement des PA pour un diamètre d'axone donné. De plus, chaque courbe représente une largeur d'impulsion de stimulation différente soit, 100, 175 et 500 µsec. Ainsi, avec une fréquence et une largeur d'impulsion données, on peut savoir quels axones vont déclencher des PA et ceux qui seront inhibés. Les courbes montrent pour chacun des diamètres d'axones, la fréquence nécessaire pour inhiber la propagation des PA. Par exemple, avec ces courbes on peu trouver la combinaison fréquence – largeur d'impulsion nécessaire pour inhiber les PA à l'intérieur des fibres possédant un diamètre proche de 12,8 µm : 1500 Hz avec une

largeur de 100 µs. Imaginons que nous traçons une ligne horizontal sur ce point, en augmentant la fréquence de stimulation, on déplace notre ligne imaginaire vers le haut jusqu'à ce qu'elle croise la courbe pour les diamètres d'axones de 11,5 et 14 µm. Ainsi, avec une combinaison fréquence – largeur d'environ 1575 Hz et 100 µs, on inhibe la propagation des PA pour les diamètres d'environ 11,5 µm à 14 µm inclusivement.

Il est maintenant possible d'observer que la largeur d'impulsion est un paramètre non négligeable. Les courbes de sélectivité varient beaucoup en fonction de la largeur d'impulsion; ainsi, lorsqu'on l'augmente, les fréquences nécessaires pour inhiber certains diamètres d'axone sont plus faibles, cependant, un plus grand nombre d'axones seront affectés par une même fréquence.

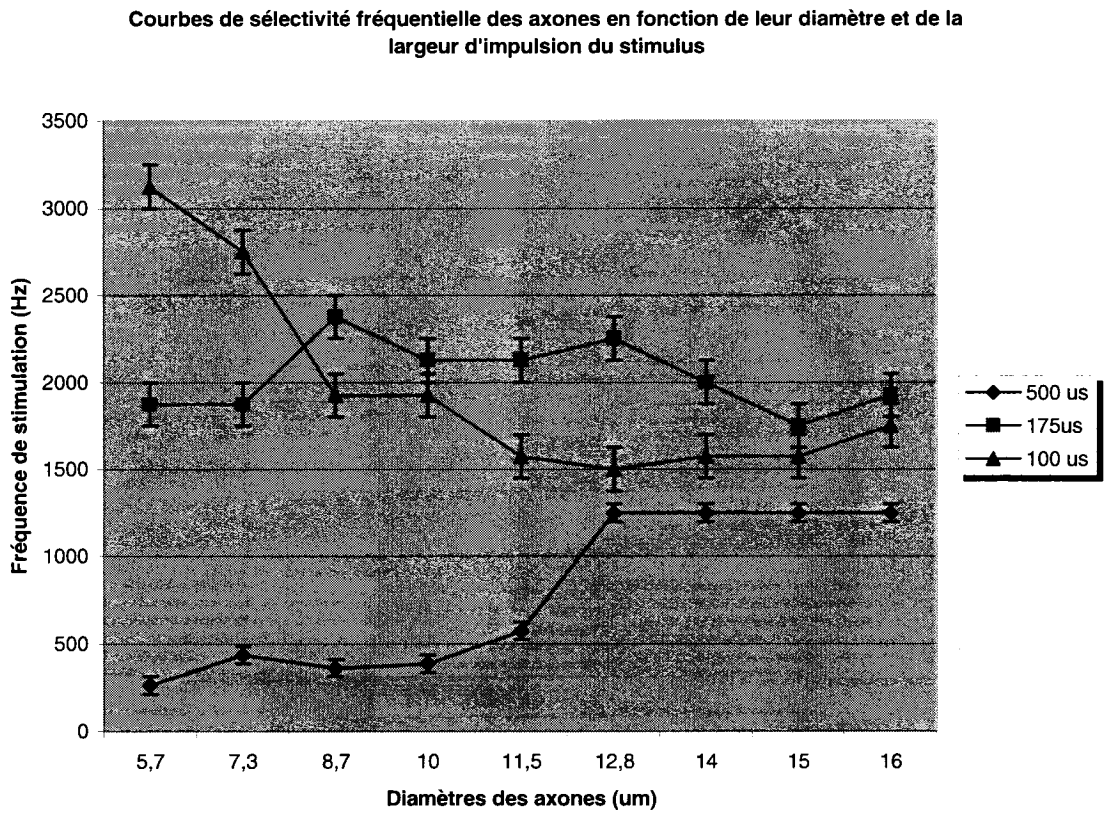


Figure 5-5 : Courbes de sélectivité : seuil fréquentiel afin d'inhiber la propagation des PA pour chacun des diamètres d'axones

Le plafonnement observé pour la courbe représentant une largeur de 500 µs provient du fait que les axones de diamètre élevé répondent plus ou moins de la même manière. En effet, avec une largeur d'impulsion élevée, la différence de fréquence nécessaire pour inhiber le déclenchement des PA est négligeable. Par contre, plus on diminue la largeur des impulsions, plus les différences entre les fibres de gros diamètres apparaissent. On remarque également que les « Courbes de sélectivités » présentées pour les largeurs d'impulsion de 100 et 175 µs se croisent. Ceci indique tout simplement que les petites fibres sont capables de supporter des fréquences de stimulation plus élevées au fur et à mesure que l'on diminue la largeur des impulsions. Cependant, ce n'est pas le cas pour les fibres possédants un diamètre plus élevé. Ce phénomène est probablement relié à la durée des périodes réfractaires absolue et relative ainsi qu'au nombre de charges injectées. Ce nombre est plus élevé avec une impulsion de 175 µs qu'avec une de 100 µs, il est fort probable que le nombre de charge injectés dans le cas d'une impulsion de largeur de 175 µs soit suffisant pour déclencher un PA lors de la période réfractaire relative, mais pas dans le cas d'une impulsion possédant une largeur de 100 µs.

5.4 Conclusion

Par rapport au projet de l'implant urinaire et du problème de dyssynergie, on peut déduire clairement que la stimulation « basse fréquence » utilisé de 30 Hz, déclenche des PA pour toutes les fibres. Les résultats observés lors des essais in-vivo démontre également qu'avec cette fréquence, on obtient une bonne contraction de la vessie. De par les résultats de simulation obtenus avec notre modèle, on voit également qu'une sélectivité des axones à l'aide d'un signal « haute fréquence » est également possible. Une combinaison judicieuse des deux signaux de stimulation permettrait alors d'obtenir que certains axones déclenchent des PA alors qu'on en inhiberait d'autres en même temps. Si les études histologiques corroborent la répartition des axones en fonction de leur diamètre vers les organes concernés, soit la vessie (petites fibres) et le sphincter

(grosses fibres), on peut donc stimuler la vessie tout en inhibant les PA se dirigeant vers le sphincter.

Le prochain chapitre présente une discussion sur les résultats obtenus, les limites de notre modèle ainsi que des recommandations pour des travaux futurs.

Chapitre 6
DISCUSSION ET CONCLUSION GÉNÉRALE

Les présents travaux ont permis de construire un modèle de nerf qui rend compte de phénomènes physiologiques du système nerveux périphérique (section 5-1). Notre modèle permet de simuler le comportement d'un nerf suite à l'application d'un stimulus provenant d'une électrode bipolaire extracellulaire de type « cuff » entourant le nerf. Une application a également été développée afin de faciliter la création du modèle en permettant à l'utilisateur de spécifier les différents paramètres constituants du nerf qu'il désire modéliser. Une série de fichiers créés sont alors utilisés par un logiciel d'analyse des éléments finis qui calcul la propagation du courant provenant d'un des contacts de l'électrode au travers de toutes les couches de tissus physiologiques présentes dans le nerf. Par la suite, les valeurs de potentiels présentes autour de chacun des axones sont extraites dans un fichier afin d'être utilisées par le logiciel Neuron qui simule la propagation des PA.

L'approche utilisée pour la réalisation de notre modèle est la suivante : modélisation des plus petits éléments constituant le nerf tout en remontant jusqu'à l'obtention d'un nerf complet. À la base, notre modèle est une série d'équations modélisant les différents canaux de conductions ioniques présents au niveau de la membrane des axones. Ensuite, la gaine de myéline recouvrant les axones est modélisée ainsi que le tissu que l'on retrouve à l'intérieur des fascicules. La membrane entourant les fascicules et le tissu constituant l'espace entre les différents fascicules est également modélisé. Finalement, une électrode est générée afin d'injecter des charges au moyen d'un courant électrique.

Il est important de mentionner que pour une éventuelle utilisation de notre modèle par notre équipe, tous les paramètres physiologiques du nerf et de l'électrode peuvent être modifiés. On obtient ainsi une grande versatilité du modèle et il devient possible de

générer plusieurs configurations neuronales selon nos besoins. Il est également possible de modifier les paramètres de stimulations afin de tester différentes combinaisons de fréquence, d'amplitude et de largeur d'impulsions.

Les résultats présentés soutiennent donc la faisabilité d'obtenir une sélectivité des axones au sein d'un nerf à l'aide d'un signal de stimulation qui varie dans le domaine fréquentiel. Cependant, ces mêmes résultats ont démontré l'importance de la largeur des impulsions utilisées. En effet, les courbes obtenues n'ont pas la même « configuration » dépendamment de la largeur des impulsions utilisées. Il n'y a donc pas uniquement la composante fréquentielle du signal qui permet la sélectivité, mais bien une combinaison de deux paramètres. Tel que présenté à la section 5-3, une combinaison judicieuse de la fréquence et de la largeur d'impulsion permet d'activer ou d'inhiber un certain nombre d'axones indépendamment des autres. En effet, les résultats obtenus ont permis de générer trois « Courbes de sélectivité » auxquelles nous pouvons nous référer afin de déterminer les largeurs d'impulsions ainsi que les fréquences nécessaires pour activer ou inhiber de manière sélective les axones au sein d'un nerf. Tel que mentionné précédemment, le modèle est basé sur les échanges ioniques au niveau de la membrane des axones. Ce niveau de précision élevé nous permet d'affirmer que les résultats sont très proches de la réalité physiologique. La validation effectuée démontre que le comportement de notre modèle est semblable au comportement observé lors de différentes manipulations expérimentales.

6.1 Conclusion et travaux futurs

Les valeurs de fréquences obtenues suite aux simulations peuvent sembler élevées; ce qui pourrait suggérer la possibilité de la présence d'un autre phénomène entrant en compte lors d'une stimulation électrique neuronale. Cette question demeure sans réponse définitive, cependant, nous avons pris certaines décisions, lors de la réalisation du modèle, qui peuvent influencer les résultats. Entre autre, nous avons décidé de ne pas

modéliser le phénomène d'accumulation des ions de calcium à l'extérieur de la membrane lors de la dépolarisation. Cette accumulation peut ralentir les échanges ioniques des autres canaux et ainsi modifier le temps de recouvrement de l'axone, ce qui influencerait probablement les valeurs des fréquences obtenues à la baisse. Un autre aspect important pouvant influencer les résultats provient de la plaque motrice que l'on retrouve normalement au niveau de la jonction neuromusculaire. Cette plaque transmet les influx nerveux provenant des axones aux fibres musculaires via un intermédiaire; l'acétylcholine (ACh). Cette transmission possède ses propres caractéristiques qui pourraient influencer la valeur des fréquences, non pas au niveau de la sélectivité des axones, mais plutôt au niveau de la fréquence nécessaire pour activer ou inhiber une fibre musculaire. Il est possible qu'un axone déclenche des PA à une certaine fréquence, mais que la fibre musculaire associée à cet axone ne « suive » pas la cadence, la fibre s'en retrouverait inactive, c'est le phénomène de fatigue musculaire. Il est également possible que le temps de réaction de la plaque motrice (dépolarisation et repolarisation) ait un impact.

Afin de valider ces explications, il faudrait modéliser la plaque motrice et l'ajouter à l'extrémité de notre modèle du nerf. On pourrait également ajouter le phénomène d'accumulation des ions de calcium à l'extérieur de la membrane afin d'évaluer son impact réel sur le temps de recouvrement, mais surtout sur les résultats présentés. Il est important de mentionner que les courbes présentées ont été obtenues suite à des simulations avec un stimulus possédant une amplitude de 1 mA. Afin d'obtenir des courbes de sélectivités plus exhaustives, il faudrait réaliser des simulations de propagation des PA avec une plage de valeur d'amplitude plus grande. Ceci permettrait probablement d'obtenir plus de courbes et ainsi offrir plus de choix judicieux au niveau de la combinaison fréquence et largeur d'impulsions possible.

En dernier lieu, la configuration du nerf utilisée provient d'une approximation générale extraite de la littérature. Il serait intéressant d'effectuer une série d'études

histologiques sur le nerf utilisé lors des expériences (S2 dans le cas présent) afin de mieux connaître son anatomie i.e. nombre de fascicules, nombre d'axones par fascicule, diamètre des fascicules présents et épaisseur des différents tissus. Plusieurs mesures de pression (en cm H_2O) exercé par la vessie et le sphincter ainsi que l'EMG correspondant ont été effectuées lors de l'implantation et même post-implantation au courant de ces travaux de recherches. Malheureusement, les mesures effectuées n'apportent aucune contribution précise aux présents travaux. Exception faites qu'une stimulation « basse fréquence » nous permet effectivement d'obtenir une bonne contraction au niveau de la vessie. Lors de certains essais, on pouvait observer une diminution de la pression sphinctérienne suite à l'application d'un signal combiné de haute (fréquence maximum utilisée de 650Hz) et basse fréquence, cependant, ce n'était pas suffisant pour obtenir une miction. Les résultats présentés dans cet ouvrage semblent indiquer que la valeur utilisée pour les « hautes fréquences » lors des expérimentations est trop basse.

BIBLIOGRAPHIE

[1] A.D.A.M. (2005), Health Illustrated Encyclopedia, http://www.nlm.nih.gov/medlineplus/ency/imagepages/9682.htm

[2] BAKER, M., BOSTOCK, H. et al., Function and distribution of three types of rectifying channel in rat spinal root myelinated axons. *Journal of Physiology* 383, 45-67, 1987.

[3] BOWE, C. M.,. KOCSIS, J. D et al., The association of the supernormal period and the depolarizing afterpotential in myelinated frog and rat sciatic nerve. *Neuroscience* 21: 2, 585-593, 1987.

[4] BOYER, S., SAWAN, M. et al., Implantable selective stimulator to improve bladder voiding: design and chronic experiments in dogs. *IEEE Transactions on Rehabilitation Engineering* 8: 4, 464-470, 2000.

[5] BRINDLEY, G. S., The first 500 patients with sacral anterior root stimulator implants: general description. *Paraplegia* 32: 12, 795-805, 1994.

[6] BRODAL, P., The central nervous system : structure and function. Oxford ; New York, *Oxford University Press*, 515, 2004.

[7] BUZELIN, J.-M. and AVEROUS, M. Urodynamique : bas appareil urinaire. Paris ; New York ; Barcelone, *Masson*, 200, 1984.

[8] CHIU, S. Y., RITCHIE, J. M.et al., A quantitative description of membrane currents in rabbit myelinated nerve. *Journal of Physiology* 292, 149-166, 1979.

[9] FUNG, K.-M. (2002). The neuropathology Learning program for Resident and Medical students. Oklahoma, *Department of Pathology, University of Oklahoma.*

[10] HINES, M. L. and CARNEVALE, N. T. Neuron: A tool for neuroscientists, *The Neuroscientist*, 7: 123-135, 2001, www.neuron.yale.edu.

[11] HODGKIN, A. L. and HUXLEY, A. F. A quantitative description of membrane current and its application to conduction and excitation in nerve. *Journal of Physiology* 117: 4, 500-544, 1952.

[12] Institute S. C. I., SCIRun: A scientific computing problem solving environment, http://software.sci.utah.edu/scirun.html.

[13] LI, J. S., HASSOUNA, M. et al., Electrical stimulation induced sphincter fatigue during voiding. *Journal of Urology* 148: 3, 949-952, 1992.

[14] MANN, M. D. Peripheral Nerves. The Nervous System In Action. Omaha, 1997.

[15] MCINTYRE, C. C. and GRILL, W. M. Extracellular stimulation of central neurons: influence of stimulus waveform and frequency on neuronal output. *Journal of Neurophysiology* 88: 4, 1592-1604, 2002.

[16] MCINTYRE, C. C., RICHARDSON, A. G. et al., Modeling the excitability of mammalian nerve fibers: influence of afterpotentials on the recovery cycle. *Journal of Neurophysiology* 87: 2, 995-1006, 2002.

[17] MCNEAL, D. R., Analysis of a model for excitation of myelinated nerve. *IEEE Transactions on Biomedical Engineering* 23: 4, 329-337, 1976.

[18] PEREZ-ORIVE, J. and DURAND D.M., Modeling study of peripheral nerve recording selectivity. *IEEE Transactions on Rehabilitation Engineering* 8: 3, 320-329, 2000.

[19] RATTAY, F. and ABERHAM, M., Modeling axon membranes for functional electrical stimulation. *IEEE Transactions on Biomedical Engineering* 40: 12, 1201-1209, 1993.

[20] RIJKHOFF, N. J., HOLSHEIMER, J. et al., Selective stimulation of sacral nerve roots for bladder control: a study by computer modeling. *IEEE Transactions on Biomedical Engineering* 41: 5, 413-424, 1994.

[21] RIJKHOFF, N. J., WIJKSTRA, H. et al., Urinary bladder control by electrical stimulation: review of electrical stimulation techniques in spinal cord injury. *Neurourology and Urodynamics* 16: 1, 39-53, 1997.

[22] ROBIN, S., SAWAN, M. et al., Implantable stimulation system dedicated for neural selective stimulation. *Medical and Biological Engineering and Computing* 36: 4, 490-492, 1998.

[23] SCHIEFER, M. A., GUSTAFSON, R. J. et al. Standing Neuroprosthetic: Modeling Selective stimulation with a FINE. *10th Annual Conference of the International Functionnal Electrical Stimulation Society*, Montreal, Canada, 256-258, 2005.

[24] SHAKER, H. S., TU, L. M. et al., Reduction of bladder outlet resistance by selective sacral root stimulation using high-frequency blockade in dogs: an acute study. *Journal of Urology* 160: 3 Pt 1, 901-907, 1998.

[25] STOVER, S. L., DELISA, J. A. et al., Spinal cord injury : clinical outcomes from the model systems. Gaithersburg, Md., *Aspen Publications*, 355, 1995.

[26] SWEENEY, J., MORTIMER, J. et al. Modeling of mammalian myelinated nerve for functional neuromuscular electrostimulation. *IEEE 9th Annual Conference on Engineering in Medecine and Biology Society*, Boston, MA, 1987.

[27] TAI, C., DE GROAT, W. C. et al., Simulation analysis of conduction block in unmyelinated axons induced by high-frequency biphasic electrical currents. *IEEE Transactions on Biomedical Engineering* 52: 7, 1323-1332, 2005.

[28] VELTINK, P. H., VAN VEEN, B. K. et al., A modeling study of nerve fascicle stimulation. *IEEE Transactions on Biomedical Engineering* 36: 7, 683-692, 1989.

[29] WALTER, J. S., WHELLER, J. S. et al., Dynamic bolbocavernous reflex: Dyssynergia evaluation following sci. *Journal of American Paraplegia Society* 17, 140-145, 1994.

Annexe I

CODE DU « PERIPHERAL NERVE BUILDER »

```
//*****************************************************************//
//PERIPHERAL NERVE BUILDER----------v1---------- Octobre 2005
//---includes Axon position generator and nerve finite element mesh builder
//Synopsis: The user defines fascicle position and radius, the program
//        computes position and diameter of axons inside fascicles.
//        The user can delete/undelete fascicles, view individual info,
//        create stat file, and create a NEURON (http://www.neuron.yale.edu)
//        compatible file containing axon diameter and position.
//
//        Furthermore, the user can create a Finite Element model of
//        the nerve by specifying nerve geometry, conductivity values,
//        and resolution of mesh. Additionally, the user can specify the
//        position of the cuff electrode and specify the lentgh of the section
//        where voltages are to be calculated. The mesh files are compatible with
//        the finite element analysis program SCIRun (http://www.sci.utah.edu).
//
//        SCIRun will in turn use the output files of the C++ program to
//        solve for the forward finite element problem and obtain the
//        voltages around each axon. NEURON, on the other
//        hand, will use the C++ output files to geometrically locate and
//        build the axons while inserting voltage stimulus outside each axon
//        as solved for by SCIRun.
//
//Étudiant à la maîtrise:
//          Dominique Pâquet-Ferron (dominique.paquet-ferron@polymtl.ca)
//Trainees:
//        Sleiman Bou-Sleiman (ssb08@aub.edu.lb)
//        Germain Aoun (gga07@aub.edu.lb)
//        Simon Fouquette (simfouq@hotmail.com)
//
//Institut de Génie Biomédical - École Polytechnique de Montréal - Montréal, Canada
//                  www.igb.umontreal.ca
```

```
//ECE Department - American University of Beirut - Beirut, Lebanon
//        www.aub.edu.lb
//Département d'informatique et de recherche opérationnelle (Bio-Informatique) –
// Université de Montréal - Montréal, Canada
//        www.bioinfo.umontreal.ca
//Polystim research group - Ecole Polytechnique Montreal - Montreal, Canada
//              www.polystim.polymtl.ca
//*****************************************************************//

#include <iostream.h>
#include <conio.h>
#include <windows.h>
#include <iomanip.h>
#include <stdlib.h>
#include <math.h>
#include <stdio.h>
#include <time.h>
#include <fstream.h>
#include <string.h>


//******************GLOBAL VARIABLES**********************

const int MAXFASC = 10;

int nFibD0[MAXFASC];
int nFibD1[MAXFASC];
int nFibD2[MAXFASC];
int nFibD3[MAXFASC];
int nFibD4[MAXFASC];
int nFibD5[MAXFASC];
int nFibD6[MAXFASC];
int nFibD7[MAXFASC];
int nFibD8[MAXFASC];

//fiber diameters
double fibD0 = 0;
double fibD1 = 0;
double fibD2 = 0;
double fibD3 = 0;
double fibD4 = 0;
double fibD5 = 0;
double fibD6 = 0;
```

```
double fibD7 = 0;
double fibD8 = 0;
//fiber distributions (percentages)
double pFibD0 = 0;
double pFibD1 = 0;
double pFibD2 = 0;
double pFibD3 = 0;
double pFibD4 = 0;
double pFibD5 = 0;
double pFibD6 = 0;
double pFibD7 = 0;
double pFibD8 = 0;
int fasc_num = 0;
double max_diam = 16.0; //maximum diameter used in our models
double totalaxonarea =0;
const double PI = 3.14159265358979323846;

//variables used in creating finite element meshes
int nrv_rad, xtra, perithick;
double mm = 0.001;

//variables that hold FE stats
int lay0=0, lay1=0, lay2=0, lay3=0, lay4=0;
int meas_sites = 0;

//used to create the locus file
char * buffer;
long size;

//name of temporary files holding axon positions
char tempname[16] = "/temp/fascx.txt";

//some variables used to alert the user to produce all necessary files before exiting
bool step[5];

//holds geometrical info for a fascicle
struct ALLFASC{
        double radius;
        double x;
        double y;
        bool include;
        int totalaxons;
};
ALLFASC fasc[MAXFASC];
```

```
//*********************COLORTEXT*****************************
//DESCRIPTION: changes color of text...just for display purposes!!
//*********passed parameters: int: choice of color
//*********returns: void
//************************************************************
void colorText(int option)
{
HANDLE OutputH;
OutputH = GetStdHandle(STD_OUTPUT_HANDLE);
switch(option)
{
case 0: SetConsoleTextAttribute(OutputH, FOREGROUND_RED |
FOREGROUND_GREEN | FOREGROUND_BLUE); break; //normal white
case 1: SetConsoleTextAttribute(OutputH, BACKGROUND_BLUE|
BACKGROUND_GREEN); break; //background blue
case 2: SetConsoleTextAttribute(OutputH, FOREGROUND_BLUE|
FOREGROUND_GREEN); break; //foreground blue
case 3: SetConsoleTextAttribute(OutputH, BACKGROUND_RED); break;//WARNING
case 4: SetConsoleTextAttribute(OutputH, FOREGROUND_RED); break;//warning
message
case 5: SetConsoleTextAttribute(OutputH, FOREGROUND_GREEN); break; //steps
case 6: SetConsoleTextAttribute(OutputH, BACKGROUND_GREEN); break; //steps
default: SetConsoleTextAttribute(OutputH, FOREGROUND_RED |
FOREGROUND_GREEN | FOREGROUND_BLUE); break;
}
}

//*********************FIBERDIAM*****************************
//DESCRIPTION: Selects fiber diameter (in micrometers)
//          from the prespecified values.
//*********passed parameters: none
//*********returns: double:
//              one of the following values (5.7; 7.3; 8.7;
//                    10.0; 11.5; 12.8; 14.0; 15.0; 16.0)
//              or user defined values...
//************************************************************
double fiberdiam()
{
        int prob;
        prob = rand()%100;
        if(prob<pFibD0)
        {
                nFibD0[fasc_num-1]+=1;
```

```
                return fibD0; //23%
        }
        else if(prob<pFibD0+pFibD1)
        {
                nFibD1[fasc_num-1]+=1;
                return fibD1; //22%
        }
        else if(prob<pFibD0+pFibD1+pFibD2)
        {
                nFibD2[fasc_num-1]+=1;
                return fibD2; //10%
        }
        else if(prob<pFibD0+pFibD1+pFibD2+pFibD3)
        {
                nFibD3[fasc_num-1]+=1;
                return fibD3; //10%
        }
        else if(prob<pFibD0+pFibD1+pFibD2+pFibD3+pFibD4)
        {
                nFibD4[fasc_num-1]+=1;
                return fibD4; //10%
        }
        else if(prob<pFibD0+pFibD1+pFibD2+pFibD3+pFibD4+pFibD5)
        {
                nFibD5[fasc_num-1]+=1;
                return fibD5; //10%
        }
        else if(prob<pFibD0+pFibD1+pFibD2+pFibD3+pFibD4+pFibD5+pFibD6)
        {
                nFibD6[fasc_num-1]+=1;
                return fibD6; //5%
        }
        else
if(prob<pFibD0+pFibD1+pFibD2+pFibD3+pFibD4+pFibD5+pFibD6+pFibD7)
        {
                nFibD7[fasc_num-1]+=1;
                return fibD7;//5%
        }
        else
        {
                nFibD8[fasc_num-1]+=1;
                return fibD8;//5%
        }
}
```

```
//*****************CHECK_INTERSECTION************************
//DESCRIPTION: checks if fascicle to be added or undeleted
//          intersects with another fascicle
//*********passed parameters: 1- radius;
//                    2- x-center position;
//                    3- y-center position
//*********returns: bool:
//               true if there is an intersection;
//               false if no intersection
//************************************************************

bool check_intersection(double radi, double x_cen, double y_cen)
{
        bool result = false;
        double distance; //holds distance calculated as: sqrt( (x2-x1)^2 + (y2-y1)^2 )

        for(int i=0; i<fasc_num; i++)
        {
                if(fasc[i].include==true)
                {
                distance = sqrt((x_cen - fasc[i].x)*(x_cen - fasc[i].x)+(y_cen -
fasc[i].y)*(y_cen - fasc[i].y));
                if(distance <= radi+fasc[i].radius)
                {
                        colorText(4);
                        cout<<"ERROR: New fascicle intersects fascicle number
"<<i+1<<" !!"<<endl;
                        result = true;
                }
                }
        }
        return result;
}

//*******************CREATE_FASC*****************************
//DESCRIPTION: Creates a new fascicle. User inputs radius,
//          center position in x-y plane. Function generates
//          axon positions inside the fascicle and outputs
//          to a temp file.
//*********passed parameters: none
//*********returns: void
//************************************************************
```

```
void create_fasc()
{

	double fas_radius, x_trans = 0, y_trans = 0;
	double diam = 0, x = 0, y = 0;
	bool intersect = true;


	colorText(0);
	while(intersect == true)
	{
		cout<<"Enter radius of fascicle (um): ";
		cin>>fas_radius;
		cout<<"Enter x-coordinate of fascicle center: ";
		cin>>x_trans;
		cout<<"Enter y-coordinate of fascicle center: ";
		cin>>y_trans;

		//do checking
		intersect = check_intersection(fas_radius, x_trans, y_trans);
		if(intersect == true)
			cout<<"Please change radius and/or xy coordinates!!"<<endl;
		colorText(0);

	}

	fasc_num+=1;
	nFibD0[fasc_num-1] =0; nFibD1[fasc_num-1] =0; nFibD2[fasc_num-1] =0;
	nFibD3[fasc_num-1] =0; nFibD4[fasc_num-1] =0; nFibD5[fasc_num-1] =0;
	nFibD6[fasc_num-1] =0; nFibD7[fasc_num-1] =0; nFibD8[fasc_num-1] =0;


	fasc[fasc_num-1].radius=fas_radius;
	fasc[fasc_num-1].x=x_trans;
	fasc[fasc_num-1].y=y_trans;
	fasc[fasc_num-1].include=true;
	fasc[fasc_num-1].totalaxons=0;

	char buf[1];
	itoa(fasc_num,buf,10);
	tempname[10]=buf[0];
	ofstream outfile(tempname);

	//coordinates x*x + y*y = r*r
```

```
        y = fas_radius - max_diam/2;


        while (y <= fas_radius - max_diam/2 && y >= max_diam/2 - fas_radius)
        {
                x = sqrt(pow(fas_radius,2) - pow(y,2));

                while (x >= -sqrt(pow(fas_radius,2) - pow(y,2)))
                {
                        diam = fiberdiam();
                        outfile<<diam<<" "<<x+x_trans<<" "<<y+y_trans<<endl;
                        x = x - max_diam;//maybe should be changed!!!
                        fasc[fasc_num-1].totalaxons++;
                }

                y = y - max_diam;

        }
        colorText(0);
        outfile.close();

}

//********************CREATE_STAT*****************************
//DESCRIPTION: Creates a statistics file including all fascicles
//          and their information (radius, x-y position,
//          axon diameter distribution, total number of axons
//          , total area covered by axons, and percentage of
//          fascicle area covered).
//*********passed parameters: none
//*********returns: void
//*************************************************************
void create_stat()
{
        ofstream statfile("axon_stats.txt");


        for(int i=0; i<fasc_num; i++)
        {
                if(fasc[i].include==true)
                {
                fasc[i].totalaxons = nFibD0[i]+nFibD1[i]+nFibD2[i]

        +nFibD3[i]+nFibD4[i]+nFibD5[i]
```

```
+nFibD6[i]+nFibD7[i]+nFibD8[i];

        statfile<<"FASCICLE "<<i+1<<" R="<<fasc[i].radius<<" @
("<<fasc[i].x
            <<","<<fasc[i].y<<")"<<endl;
        statfile<<" _____"<<endl;
        statfile<<"  diam(um)\tnum\tpercentage"<<endl;

        if(fibD0!=0) statfile<<"
"<<fibD0<<"\t\t"<<nFibD0[i]<<"\t"<<nFibD0[i]*100/fasc[i].totalaxons<<endl;
        if(fibD1!=0) statfile<<"
"<<fibD1<<"\t\t"<<nFibD1[i]<<"\t"<<nFibD1[i]*100/fasc[i].totalaxons<<endl;
        if(fibD2!=0) statfile<<"
"<<fibD2<<"\t\t"<<nFibD2[i]<<"\t"<<nFibD2[i]*100/fasc[i].totalaxons<<endl;
        if(fibD3!=0) statfile<<"
"<<fibD3<<"\t\t"<<nFibD3[i]<<"\t"<<nFibD3[i]*100/fasc[i].totalaxons<<endl;
        if(fibD4!=0) statfile<<"
"<<fibD4<<"\t\t"<<nFibD4[i]<<"\t"<<nFibD4[i]*100/fasc[i].totalaxons<<endl;
        if(fibD5!=0) statfile<<"
"<<fibD5<<"\t\t"<<nFibD5[i]<<"\t"<<nFibD5[i]*100/fasc[i].totalaxons<<endl;
        if(fibD6!=0) statfile<<"
"<<fibD6<<"\t\t"<<nFibD6[i]<<"\t"<<nFibD6[i]*100/fasc[i].totalaxons<<endl;
        if(fibD7!=0) statfile<<"
"<<fibD7<<"\t\t"<<nFibD7[i]<<"\t"<<nFibD7[i]*100/fasc[i].totalaxons<<endl;
        if(fibD8!=0) statfile<<"
"<<fibD8<<"\t\t"<<nFibD8[i]<<"\t"<<nFibD8[i]*100/fasc[i].totalaxons<<endl;

        statfile<<"\n\tTOTAL: "<<fasc[i].totalaxons<<endl;

        totalaxonarea =
PI*0.000001*0.000001*(nFibD0[i]*5.7*5.7+nFibD1[i]*7.3*7.3+nFibD2[i]

    *8.7*8.7+nFibD3[i]*10*10+nFibD4[i]*11.5*11.5+nFibD5[i]*12.8*12.8+nFibD6
[i]*14*14
            +nFibD7[i]*15*15+nFibD8[i]*16*16)/4;

        statfile<<"\tTOTAL AREA: "<<totalaxonarea<<" um^2"<<endl;

        statfile<<"\tAXON AREA/FASCICLE AREA:
"<<totalaxonarea/(PI*0.000001*0.000001*fasc[i].radius*fasc[i].radius/2)*100<<endl;
        statfile<<"*****************************************"<<endl;
```

```
                }
        }

        statfile.close();

}

//********************VIEWSTATS*******************************
//DESCRIPTION: Shows the information related to a specific
//          fascicle (radius, x-y position,
//          axon diameter distribution, total number of axons
//          , total area covered by axons, and percentage of
//          fascicle area covered).
//*********passed parameters: int:
//                    number of fascicle to view info
//*********returns: void
//*************************************************************

void viewstats(int choi)
{
        colorText(6);
        cout<<"FASCICLE        "<<choi+1<<"        R="<<fasc[choi].radius<<"        @
("<<fasc[choi].x<<","<<fasc[choi].y<<")"<<endl;
        colorText(5);
        cout<<"  _____"<<endl;
        cout<<"  diam(um)\tnum\tpercentage"<<endl;


        if(fibD0!=0)                                                    cout<<"
"<<fibD0<<"\t\t"<<nFibD0[choi]<<"\t"<<nFibD0[choi]*100/fasc[choi].totalaxons<<end
l;
        if(fibD1!=0)                                                    cout<<"
"<<fibD1<<"\t\t"<<nFibD1[choi]<<"\t"<<nFibD1[choi]*100/fasc[choi].totalaxons<<end
l;
        if(fibD2!=0)                                                    cout<<"
"<<fibD2<<"\t\t"<<nFibD2[choi]<<"\t"<<nFibD2[choi]*100/fasc[choi].totalaxons<<end
l;
        if(fibD3!=0)                                                    cout<<"
"<<fibD3<<"\t\t"<<nFibD3[choi]<<"\t"<<nFibD3[choi]*100/fasc[choi].totalaxons<<end
l;
        if(fibD4!=0)                                                    cout<<"
"<<fibD4<<"\t\t"<<nFibD4[choi]<<"\t"<<nFibD4[choi]*100/fasc[choi].totalaxons<<end
l;
```

```
        if(fibD5!=0)                                                    cout<<"
"<<fibD5<<"\t\t"<<nFibD5[choi]<<"\t"<<nFibD5[choi]*100/fasc[choi].totalaxons<<end
l;
        if(fibD6!=0)                                                    cout<<"
"<<fibD6<<"\t\t"<<nFibD6[choi]<<"\t"<<nFibD6[choi]*100/fasc[choi].totalaxons<<end
l;
        if(fibD7!=0)                                                    cout<<"
"<<fibD7<<"\t\t"<<nFibD7[choi]<<"\t"<<nFibD7[choi]*100/fasc[choi].totalaxons<<end
l;
        if(fibD8!=0)                                                    cout<<"
"<<fibD8<<"\t\t"<<nFibD8[choi]<<"\t"<<nFibD8[choi]*100/fasc[choi].totalaxons<<end
l;
        colorText(6);
        cout<<"\n\tTOTAL: "<<fasc[choi].totalaxons<<endl;

        totalaxonarea                                                   =
PI*0.000001*0.000001*(nFibD0[choi]*5.7*5.7+nFibD1[choi]*7.3*7.3+nFibD2[choi]*8.
7*8.7

        +nFibD3[choi]*10*10+nFibD4[choi]*11.5*11.5+nFibD5[choi]*12.8*12.8

        +nFibD6[choi]*14*14+nFibD7[choi]*15*15+nFibD8[choi]*16*16)/4;

        cout<<"\tTOTAL AREA: "<<totalaxonarea<<" um^2"<<endl;

        cout<<"\tAXON                AREA/FASCICLE                AREA:
"<<totalaxonarea/(PI*0.000001*0.000001*fasc[choi].radius*fasc[choi].radius/2)*100<<e
ndl;
        colorText(0);
}

//********************DEL_VIEW_FASC*****************************
//DESCRIPTION: Shows list of all fascicles and some of their
//          information; used for two purposes: to (un)delete
//          a fascicle and to view extended information (by
//          calling another function: viewstats(fascnum)).
//*********passed parameters: int:
//                    0 to delete a fascicle
//                    1 to view stats
//*********returns: void
//***************************************************************
void del_view_fasc(int choi)
{
        int numtodel=fasc_num+1;
```

```
        system("cls");
        colorText(6);
        cout<<"FASCICLE INFO***************"<<endl;
        colorText(5);
        cout<<"n
"<<setw(6)<<"Radius\t"<<setw(10)<<"Position\t"<<setw(5)<<"Axons\t\t\tDELETED?\n
"<<endl;
        colorText(0);
        for(int i=0; i<fasc_num; i++)
        {
                cout<<i+1<<"
"<<setw(6)<<fasc[i].radius<<"\t("<<setw(4)<<fasc[i].x<<setw(1)<<","<<setw(4)<<fasc[
i].y<<")\t"
                        <<setw(5)<<fasc[i].totalaxons;
                if(fasc[i].include==true) cout<<"\t\t\tNO"<<endl;
                else cout<<"\t\t\tYES"<<endl;
        }

        if(choi==0) //delete
        {
                while (numtodel>fasc_num || numtodel<1)
                {
                cout<<"Choose number of fascicle to delete/undelete: ";
                cin>>numtodel;
                }
                if(fasc[numtodel-1].include==false) //if fascicle is to be undeleted, first
check if it intersects other fascicles
                {
                        if(check_intersection(fasc[numtodel-1].radius, fasc[numtodel-1].x,
fasc[numtodel-1].y) == false)
                                fasc[numtodel-1].include=true;
                        else
                        {   colorText(4);
                                cout<<"Cannot undelete fascicle with an intersecting
existing fascicle!!\n"
                                        <<"HINT: To include the current fascicle, try
deleting the other one..."
                                        <<"\nPress any key to continue..."<<endl;
                                colorText(0);
                                getch();
                        }
                }
                else fasc[numtodel-1].include=false;
```

```
        }
        else //just show info
        {
                while (numtodel>fasc_num || numtodel<1)
                {
                cout<<"Choose fascicle number to view stats: ";
                cin>>numtodel;
                }
                viewstats(numtodel-1);

                cout<<"Press any key to continue..."<<endl;
                getch();
        }

}

//********************MERGE_FILES*****************************
//DESCRIPTION: Merges the temporary files created for each
//          fascicle into a unified file containing the
//          total number of axons, each axon's diameter and
//          position for all fascicles. The output file will
//          be used with NEURON.
//*********passed parameters: none
//*********returns: void
//*************************************************************
void merge_files()
{
        char buf[1];
        ifstream infile;

        ofstream outfile;


        int counter = 0; //counts how many fascicles are to be included
        int i;

        for(i = 0; i < fasc_num; i++)
   {
                if(fasc[i].include==true)
                {
                        counter++;
                }
        }
        outfile.open("locus.txt");
```

```
        outfile<<counter<<endl; //add the number of included fascicles to the file

        for(i=0; i<fasc_num; i++)
        {
                if(fasc[i].include==true) //omit deleted fascicles
                {
                        itoa(i+1,buf,10);
                        tempname[10]=buf[0];
                        infile.open(tempname,ifstream::binary);

                        // get size of file
                        infile.seekg(0,ifstream::end);
                        size=infile.tellg();
                        infile.seekg(0);

                        // allocate memory for file content
                        buffer = new char [size];

                        // read content of infile
                        infile.read (buffer,size);

                        // write to outfile but add first number of axons in fascicle
                        outfile<<fasc[i].totalaxons<<endl;
                        outfile.write (buffer,size);

                        // release dynamically-allocated memory
                        delete[] buffer;
                        infile.close();

                }
        }
        outfile.close();
}

//************************INSIDE_FASC**********************
//DESCRIPTION: checks if the mesh node is inside a fascicle
//              and if it is in the perineurium or inside the
//         fascicle (in endoneurium or an axon maybe).
//*********passed parameters: 1 - x-position of mesh node
//                     2 - y-position of mesh node
//                     3 - perineurium thickness
//*********returns: bool: true if inside a fascicle
//***********************************************************
```

```
bool inside_fasc(double x, double y, int peri_thick = 0)
{
	double dist=0;
	for(int i=0; i<fasc_num; i++)
	{
		if(fasc[i].include==true) //if fascicle included do checking
		{
			dist = sqrt(pow((x-fasc[i].x),2)+pow((y-fasc[i].y),2));
			if(dist < (fasc[i].radius + peri_thick))
				return true;
		}
	}
	return false;
}

//************************ASSIGN_COND**********************
//DESCRIPTION: assigns conductivity values to volume elements.
//*********passed parameters: 1 - x-position of mesh node
//                   2 - y-position of mesh node
//*********returns: int: conductivity index
//                 0 = extracellular conductivity
//                 1 = epineurium conductivity
//                 2 = perimeurium conductivity
//                 3 = intrafascicular conductivity
//                 4 = outside conductivity (???)
//***********************************************************

int assign_cond(double x, double y)
{
	double dist = sqrt(pow(x,2)+pow(y,2));
	int conduct;
	if(dist< nrv_rad + xtra)
	{
		conduct = 0;
		if(dist<nrv_rad)
		{
			conduct = 1;
			if(inside_fasc(x,y,perithick)==true)
			{
				conduct = 2;
				if(inside_fasc(x,y)==true)
				{
					conduct = 3;
```

```
                        }
                    }
                }
        }
        else{ conduct = 4;}

        if(conduct==0){ lay0++;}
        else if(conduct==1){ lay1++;}
        else if(conduct==2){ lay2++;}
        else if(conduct==3){ lay3++;}
        else  {  lay4++;}

        return conduct;


}

//****************DEFINE_COND*******************************
//DESCRIPTION: defines conductivity values for all layers
//          found inside a nerve (extraneural, epineurium
//          perineurium, intrafascicular)...
//*********passed parameters: none
//*********returns: void
//*********************************************************

void define_cond()
{
        double condvalue = 0;
        char answer = 'k';
        ofstream cond_table("mesh_cond_table.txt");//tensor file to be used in BIOFEM

        while (answer != 'y' && answer!='n' && answer!='Y' && answer!='N')
        {
                cout<<"Do you want to use default conductivities? (y/n) ";
                cin>> answer;
        }

        if(answer == 'n' || answer == 'N')
        {
                cond_table<<5<<" "<<9<<endl;

                cout<<"1 - Conductivity of extraneural space (mho/m)\n\t1.1 - x: ";
                cin >> condvalue;
                cond_table<<condvalue<<" 0.0 0.0 0.0 ";
```

```
cout<<"\t1.2 - y: ";
cin >> condvalue;
cond_table<<condvalue<<" 0.0 0.0 0.0 ";
cout<<"\t1.3 - z: ";
cin >> condvalue;
cond_table<<condvalue<<endl;

cout<<"2 - Conductivity of epineurium (mho/m)\n\t2.1 - x: ";
cin >> condvalue;
cond_table<<condvalue<<" 0.0 0.0 0.0 ";
cout<<"\t2.2 - y: ";
cin >> condvalue;
cond_table<<condvalue<<" 0.0 0.0 0.0 ";
cout<<"\t2.3 - z: ";
cin >> condvalue;
cond_table<<condvalue<<endl;

cout<<"3 - Conductivity of perineurium (mho/m)\n\t3.1 - x: ";
cin >> condvalue;
cond_table<<condvalue<<" 0.0 0.0 0.0 ";
cout<<"\t3.2 - y: ";
cin >> condvalue;
cond_table<<condvalue<<" 0.0 0.0 0.0 ";
cout<<"\t3.3 - z: ";
cin >> condvalue;
cond_table<<condvalue<<endl;

cout<<"4 - Conductivity of intrafascicle (mho/m)\n\t4.1 - x: ";
cin >> condvalue;
cond_table<<condvalue<<" 0.0 0.0 0.0 ";
cout<<"\t4.2 - y: ";
cin >> condvalue;
cond_table<<condvalue<<" 0.0 0.0 0.0 ";
cout<<"\t4.3 - z: ";
cin >> condvalue;
cond_table<<condvalue<<endl;

cout<<"5 - Conductivity of surface elements (mho/m)\n\t5.1 - x: ";
cin >> condvalue;
cond_table<<condvalue<<" 0.0 0.0 0.0 ";
cout<<"\t5.2 - y: ";
cin >> condvalue;
cond_table<<condvalue<<" 0.0 0.0 0.0 ";
cout<<"\t5.3 - z: ";
```

```
                cin >> condvalue;
                cond_table<<condvalue<<endl;
        }
        else //from P.H. Veltink et al., A Modeling Study of Nerve Fascicle Stimulation
                //IEEE Trans. Bio. Eng. vol.36 no. 7, July 1989
        {
                cond_table<<5<<" "<<9<<endl;
                cond_table<<"0.04 0.0 0.0 0.0 0.04 0.0 0.0 0.0 0.04"<<endl;//human fat
tissue
                cond_table<<"0.10 0.0 0.0 0.0 0.10 0.0 0.0 0.0 0.10"<<endl;//epineurium
                cond_table<<"0.01 0.0 0.0 0.0 0.01 0.0 0.0 0.0 0.01"<<endl;//perineurium
                cond_table<<"0.08 0.0 0.0 0.0 0.08 0.0 0.0 0.0 0.08"<<endl;//fascicle
                cond_table<<"0.0001     0.0     0.0     0.0     0.0001     0.0     0.0     0.0
0.0001"<<endl;//surface tissue?

        }
        cond_table.close();


}

//********************CREATE_ELECTRODE***********************
//DESCRIPTION: creates a point cloud mesh of cuff electrodes
//*********passed parameters: none
//*********returns: void
//************************************************************

void create_electrode()
{
        ofstream elect_cathode("cuff_cath.pts");
        ofstream elect_anode("cuff_anod.pts");

        double elect_rad, y, x,z, z_start, elect_res, elect_curr, elect_sep, elect_width,
z_res;
        int elect_pts=0;

        cout<<"Electrode radius (mm): ";
        cin>>elect_rad;

        cout<<"Electrode seperation (mm): ";
        cin>>elect_sep;

        cout<<"Electrode current amplitude (A): ";
        cin>>elect_curr;
```

```
	cout<<"Building point cloud mesh:\n\tElectrode mesh resolution: ";
	cin>>elect_res;
	//elect_res = 0.05;

	cout<<"\tElectrode starting z-depth (mm): ";
	cin>>z_start;

	cout<<"\tElectrode width (mm): ";
	cin >> elect_width;

	cout<<"\tElectrode  z-resolution (mm): ";
	cin >> z_res;

	//z_res = 0.1;
	cout<<"elect_pts=  "<<elect_pts;

	y=elect_rad;
	while(y>-elect_rad)
	{
		elect_pts++; y-=elect_res;
	}

	elect_cathode<<((elect_width / z_res)+1) * (2*elect_pts-1) <<endl;
	elect_anode<<((elect_width / z_res)+1) * (2*elect_pts-1) <<endl;

	for(z=z_start; z<=z_start+elect_width+z_res; z+=z_res){
		cout << "z=  " << z<<endl;
		y=elect_rad;
		while(y>-elect_rad)
		{
			x=sqrt(pow(elect_rad,2)-pow(y,2));
			elect_cathode<<x<<" "<<y<<" "<<-z<<endl;
			elect_anode<<x<<"		"<<y<<"		"<<(-z-elect_sep-
elect_width)<<endl;

			if(x!=0)
			{
			elect_cathode<<-x<<" "<<y<<" "<<-z<<endl;
			elect_anode<<-x<<"		"<<y<<"		"<<(-z-elect_sep-
elect_width)<<endl;
			}
			y-=elect_res;
```

```
                }
        }
        elect_anode.close();
        elect_cathode.close();

        //build matrix containing current amplitude for cathode
        elect_cathode.open("cuff_curr.txt");
        elect_cathode<<((elect_width / z_res)+1) * (2*elect_pts-1)<<endl;

        for(int i=0; i<((elect_width / z_res)+1) * (2*elect_pts-1); i++)
                elect_cathode<<(elect_curr/(((elect_width / z_res)+1) * (2*elect_pts-
1)))<<endl;

        elect_cathode.close();

}
//********************CREATE_READING_ELECTRODE*********************
***
//DESCRIPTION:
//*********passed parameters: none
//*********returns: void
//************************************************************

void create_reading_electrode()
{
        ofstream elect1("reading_elect1.pts");
        ofstream elect2("reading_elect2.pts");
        ofstream elect3("reading_elect3.pts");

        double elect_rad, y, x,z, z_start, elect_res, elect_sep1, elect_sep2, elect_width,
z_res;
        int elect_pts=0;

        cout<<"Electrode radius (mm): ";
        cin>>elect_rad;

        cout<<"First electrode seperation (mm): ";
        cin>>elect_sep1;


        cout<<"Second electrode seperation (mm): ";
        cin>>elect_sep2;
```

```
	cout<<"Building point cloud mesh:\n\tElectrode mesh resolution: ";
	cin>>elect_res;
	//elect_res = 0.05;

	cout<<"\tElectrode starting z-depth (mm): ";
	cin>>z_start;

	cout<<"\tElectrode width (mm): ";
	cin >> elect_width;

	cout<<"\tElectrode  z-resolution (mm): ";
	cin >> z_res;
	//z_res = 0.1;

	y=elect_rad;
	while(y>-elect_rad)
	{
		elect_pts++; y-=elect_res;
	}

	elect1<<((elect_width / z_res)+1) * (2*elect_pts-1) <<endl;
	elect2<<((elect_width / z_res)+1) * (2*elect_pts-1) <<endl;
	elect3<<((elect_width / z_res)+1) * (2*elect_pts-1) <<endl;

	for(z=z_start; z<=z_start+elect_width+z_res; z+=z_res)
 {
		y=elect_rad;
		while(y>-elect_rad)
		{
			x=sqrt(pow(elect_rad,2)-pow(y,2));
			elect1<<x<<" "<<y<<" "<<-z<<endl;
			elect2<<x<<" "<<y<<" "<<(-z-elect_sep1-elect_width)<<endl;
			elect3<<x<<" "<<y<<" "<<(-z-elect_sep1-elect_sep2-elect_width-
elect_width)<<endl;
			if(x!=0)
			{
			elect1<<-x<<" "<<y<<" "<<-z<<endl;
			elect2<<-x<<" "<<y<<" "<<(-z-elect_sep1-elect_width)<<endl;
			elect3<<-x<<"	"<<y<<"	"<<(-z-elect_sep1-elect_sep2-
elect_width-elect_width)<<endl;

			}
			y-=elect_res;
		}}
```

```
	elect1.close();
	elect2.close();
	elect3.close();
}
//************************CREATE_MESH**********************
//DESCRIPTION: creates the 3d mesh. asks the user to input
//		some geometrical information regarding nerve
//		radius, extraneural space, perineural thickness,
//		and the resolution of the mesh in the x,y,z
//		domain.
//*********passed parameters: none
//*********returns: void
//************************************************************
void create_mesh()
{
	double x_res, y_res, z_res, x, y, z, z_depth;
	int vol_elts=0, NI = 0, NJ = 0, NK = 0;
	double condvalue = 0;
	char mesh_ans='k';
	lay0=0; lay1=0; lay2=0; lay3=0; lay4=0;

	colorText(5);
	cout<<"STEP I: DEFINE CONDUCTIVITY VALUES"<<endl;
	colorText(0);
	define_cond();
	cout<<"Conductivity table created!!!"<<endl;

	colorText(5);
	cout<<"STEP II: DEFINE GEOMETRY"<<endl;
	colorText(0);
	//maybe we need to implement checking!!

		while(mesh_ans!='h' && mesh_ans!='H' && mesh_ans!='t' &&
mesh_ans!='T')
		{
			cout<<"Do you want to create a hexahedral (h) or tetrahedral (t)
mesh? : ";
			cin>>mesh_ans;
		}
		cout<<"Enter radius of nerve (um): ";
		cin>>nrv_rad;
		cout<<"Enter extra thickness around nerve (um): ";
		cin>>xtra;
		cout<<"Enter perineurium thickness (um): ";
```

```
		cin>>perithick;
		cout<<"Enter z-depth (um): ";
		cin>>z_depth;
		cout<<"Enter x-resolution (um): ";
		cin>>x_res;
		cout<<"Enter y-resolution (um): ";
		cin>>y_res;
		cout<<"Enter z-resolution (um): ";
		cin>>z_res;

	ofstream tempfile1("/temp/temp_mesh_pts.txt");
	ofstream tempfile2;


	x = -nrv_rad - xtra;
	y = nrv_rad + xtra;
	z = 0;
	int conduct_result;
	int hexdiv = 1;

	if(mesh_ans=='t'||mesh_ans=='T')
	{
		tempfile2.open("/temp/temp_mesh_index_tet.txt");
		hexdiv=6;
	}
	else
{
		tempfile2.open("/temp/temp_mesh_index_hex.txt");
}

	while(z >= -z_depth)
	{
		NK++;
		while(y >= -nrv_rad-xtra)
		{
			NJ++;
			while(x <= nrv_rad+xtra)
			{
				NI++;
				//write coordinates of hexahedral nodes
				tempfile1<<x*mm<<" "<<y*mm<<" "<<z*mm<<endl;

		//write associated conductivity index to hexahedral volume element
	if(x<=nrv_rad+xtra-x_res && y>=-nrv_rad-xtra+y_res && z>=-z_depth+z_res)
```

```
                        {
                                conduct_result=assign_cond(x,y);
                                for(int i=0; i<hexdiv; i++)
        {
                                        tempfile2<<conduct_result<<endl;}
                        }


                        x += x_res; //move along the axis
                }

                x = -nrv_rad - xtra; //reset x
                y -= y_res; //move down the axis
        }

        y = nrv_rad + xtra; //reset y
        z -= z_res; //move down the axis
    }

    tempfile1.close();
    tempfile2.close();

    vol_elts = (NI/NJ -1)*(NJ/NK -1)*(NK -1)*hexdiv;

//*****START CREATING THE MESH FILES TO BE USED WITH****
//************************BIOFEM********

    //create file containing all indices of (hex or tet) volume elements
    ofstream mesh_file;

    if(hexdiv==1)
  {
        mesh_file.open("mesh_hex.hex");
        }
    else
  {
        mesh_file.open("mesh_tet.tet");
        }

    int base = 0;

                mesh_file<<vol_elts<<endl;

                for(int k=0;k<NK-1;k++)
```

```
                {
                        for(int j=0; j<(NJ/NK)-1; j++)
                        {
                                for(int i=0; i<(NI/NJ)-1; i++)
                                {
                                        //hexahedral mesh creation
                                        if(hexdiv==1)
                                        {
mesh_file<<base<<"      "<<base+NI/NK<<"      "<<base+NI/NK+NI/NJ<<"      "
<<base+NI/NJ<<"      "<<base+1<<"      "<<base+NI/NK+1<<"      "
<<base+NI/NK+NI/NJ+1<<" "<<base+NI/NJ+1<<endl;
                                        }

                                        //tetrahedral mesh
                                        else if(hexdiv==6)
                                        {
mesh_file<<base+NI/NJ+NI/NK<<"      "<<base+NI/NJ<<"      "<<base+NI/NK+1
<<" "<<base+NI/NK+NI/NJ+1<<endl;

mesh_file<<base+NI/NJ<<"      "<<base+NI/NK+1<<"      "<<base+NI/NK+NI/NJ+1
<<" "<<base+NI/NJ+1<<endl;

mesh_file<<base<<" "<<base+NI/NJ<<" "<<base+1<<" "<<base+NI/NK+1<<endl;

mesh_file<<base<<"   "<<base+NI/NK<<"   "<<base+NI/NK+1<<"   "<<base+NI/NJ
<<endl;

mesh_file<<base+NI/NK<<"      "<<base+NI/NK+NI/NJ<<"      "<<base+NI/NJ
<<" "<<base+NI/NK+1<<endl;

mesh_file<<base+1<<"      "<<base+NI/NK+1<<"      "<<base+NI/NJ+1<<"      "
<<base+NI/NJ<<endl;
                                        }
                                        base+=1;
                                }
                                base+=1;
                        }
                base+=(NI/NJ);
                }
                mesh_file.close();
```

```
//create file containing all point coordinates
mesh_file.open("mesh_tet.pts");

                ifstream infile("/temp/temp_mesh_pts.txt",ifstream::binary);
                char *buff;
                long bufsize;

                // get size of file
                infile.seekg(0,ifstream::end);
                bufsize=infile.tellg();
                infile.seekg(0);

                // allocate memory for file content
                buff = new char [bufsize];

                // read content of infile
                infile.read (buff,bufsize);

                // write to outfile but add first number of points in mesh
mesh_file<<NI<<endl;
                mesh_file.write(buff,bufsize);

                // release dynamically-allocated memory
                delete[] buff;
                infile.close();
                mesh_file.close();

//create file containing mesh conductivities

std::ifstream is;
std::ofstream os;

if(hexdiv==1)
                {
                is.open("/temp/temp_mesh_index_hex.txt");
                os.open("mesh_cond_hex.con");

                }
                else
                {
                is.open("/temp/temp_mesh_index_tet.txt");
                os.open("mesh_cond_tet.con");

                }
```

```
                os<<vol_elts<<endl;
            os<<is.rdbuf();

                is.close();
                os.close();

//*************END CREATING MESH FILES TO BE USED WITH
//*****************************BIOFEM*********

//*******STAT FILE FOR FINITE ELEMENT MESH
  fstream mesh_stat("mesh_stat.txt");
      mesh_stat<<"FINITE ELEMENT MESH STATISTICS\n\nTotal number of
volume elements: "<<vol_elts;
      if(hexdiv==1){  mesh_stat<<"  hexahedra"<<endl;}  else{  mesh_stat<<"
tetrahedra"<<endl;}
      mesh_stat<<"\tLAYER\t\tVOL.ELT.NUM\tPERCENTAGE\n\t_____________
_____________________________\n"

      <<"\tSurface\t\t"<<lay4*hexdiv<<"\t\t"<<(lay4*hexdiv*100/vol_elts)<<endl;
      mesh_stat<<"\tFat\t\t"<<lay0*hexdiv<<"\t\t"<<(lay0*hexdiv*100/vol_elts)<<"\n\
tEpineurium\t"<<lay1*hexdiv<<"\t\t"<<(lay1*hexdiv*100/vol_elts)<<endl;
      mesh_stat<<"\tPerineurium\t"<<lay2*hexdiv<<"\t\t"<<(lay2*hexdiv*100/vol_elts
)<<"\n\tFascicle\t"<<lay3*hexdiv<<"\t\t"<<(lay3*hexdiv*100/vol_elts)<<endl;
      mesh_stat.close();

      //*******STAT DISPLAY FOR FINITE ELEMENT MESH
      cout<<"\nFINITE ELEMENT MESH STATISTICS\n\nTotal number of volume
elements: "<<vol_elts;
      if(hexdiv==1){cout<<" hexahedra"<<endl;} else{cout<<" tetrahedra"<<endl;}
      cout<<"\tLAYER\t\tVOL.ELT.NUM\tPERCENTAGE\n\t_____________
_____________________________\n"

      <<"\tSurface\t\t"<<lay4*hexdiv<<"\t\t"<<(lay4*hexdiv*100/vol_elts)<<endl;
      cout<<"\tFat\t\t"<<lay0*hexdiv<<"\t\t"<<(lay0*hexdiv*100/vol_elts)<<"\n\tEpin
eurium\t"<<lay1*hexdiv<<"\t\t"<<(lay1*hexdiv*100/vol_elts)<<endl;
      cout<<"\tPerineurium\t"<<lay2*hexdiv<<"\t\t"<<(lay2*hexdiv*100/vol_elts)<<"\
n\tFascicle\t"<<lay3*hexdiv<<"\t\t"<<(lay3*hexdiv*100/vol_elts)<<endl;
      cout<<"\nPress any key to continue...."<<endl;
      getch();
}
```

```
//********************** ADD_MEASUREMENT_SITES*******************
//DESCRIPTION: asks the user to enter at which depths of the
//          nerve measurements of voltages will be
//          calculated. User can enter several sites.
//**********passed parameters: none
//**********returns: void
//************************************************************
void add_measurement_sites()
{
        double x,y,z=99999999, bound;
        double z_start, z_end, increment;

        char filename[] = "axonpos.pts";
        char dummyname[] = "axondummy.cmat";
        char matrixfilename[] = "increment.txt";
        double allinallaxons = 0,  z_nb =0;


        cout << "Enter z starting point (in mm): ";
        cin>>z_start;
        cout << "Enter z ending point (in mm): ";
        cin >>z_end;
        cout << "Enter z increment (in mm): ";
        cin >> increment;

        ofstream outfile(filename);
        ofstream dummyfile(dummyname);

   int i;
        for(i=0; i<fasc_num; i++){
                if(fasc[i].include==true)
                        allinallaxons+=fasc[i].totalaxons;
        }
        z_nb = ((z_end-z_start)/increment)+1;


        double n = allinallaxons * z_nb;
        //create dummy file
        dummyfile<< n <<endl;
        for (i=0; i<n; i++)
                dummyfile<<0<<endl;
        dummyfile.close();
        //create outfile
        outfile<< n <<endl;
```

```
        for( z=z_start; z<z_end+increment; z+=increment )
        {
                for(i=0; i<fasc_num; i++)
                {
                        if(fasc[i].include==true)
                        {
                                bound = fasc[i].radius -max_diam/2;
                                y=bound;
                                while(y<=bound && y>=-bound)
                                {
                                        x=sqrt(pow(fasc[i].radius,2) - pow(y,2));

                                        while (x >= -sqrt(pow(fasc[i].radius,2) - pow(y,2)))
                                        {
                                                outfile<<(x+fasc[i].x)*mm<<"
"<<(y+fasc[i].y)*mm<<" "<<-z<<endl;
                                                x = x -max_diam;
                                        }
                                        y = y - max_diam;
                                }
                        }
                }
        }
        outfile.close();
        //create measurementSiteMatrix
        ofstream matrixfile(matrixfilename);
        matrixfile<<z_nb<<endl;
        matrixfile<<z_start<<endl;
        matrixfile<<z_start+increment<<endl;
        matrixfile.close();

        colorText(3);
        cout<<"\nMeasurement file created!!!"<<endl;
        cout<<"Dummy file created!!!"<<endl;
        cout<<"Matrix file created!!!"<<endl;
}
```

```
//**********************DEFINE_DIST*****************************
//DESCRIPTION: allows user to enter personalized diameters and
//          distributions. Up to 8 entries may be entered.
//          WARNING: unless NEURON code modified, the user
//                    cannot enter personalized data...
//*********passed parameters: none
//*********returns: void
//*************************************************************
void define_dist()
{
int numadded = 0; char ans = 'p';
while( ans!='a' && ans!='x' && ans!='A' && ans!='X')
{
        colorText(5);
        cout<<"Add fiber information (a)/ Exit (x)? ";
        cin>>ans;
        colorText(0);

        if(ans=='x'||ans=='X') return;
        else if(ans=='a'||ans=='A')
        {
                cout<<"Diameter "<<numadded+1<<": ";
                switch(numadded)
                {
                case 0: cin>>fibD0; cout<<"Percentage 1: "; cin>>pFibD0; break;
                case 1: cin>>fibD1; cout<<"Percentage 2: "; cin>>pFibD1; break;
                case 2: cin>>fibD2; cout<<"Percentage 3: "; cin>>pFibD2; break;
                case 3: cin>>fibD3; cout<<"Percentage 4: "; cin>>pFibD3; break;
                case 4: cin>>fibD4; cout<<"Percentage 5: "; cin>>pFibD4; break;
                case 5: cin>>fibD5; cout<<"Percentage 6: "; cin>>pFibD5; break;
                case 6: cin>>fibD6; cout<<"Percentage 7: "; cin>>pFibD6; break;
                case 7: cin>>fibD7; cout<<"Percentage 8: "; cin>>pFibD7; break;
                case 8: cin>>fibD8; cout<<"Percentage 9: "; cin>>pFibD8; break;
                default: break;
                }
                numadded++;
        }//end of else if

        ans = 'p'; //just to stay in the while loop

}//end of while
}//end of procedure
```

```
//*********************DEFAULT_DIST*************************
//DESCRIPTION: sets default fiber diameters and distributions
//*********passed parameters: none
//*********returns: void
//**********************************************************
void default_dist()  //diameters after McIntyre et al.
{
fibD0 = 5.7; pFibD0 = 10;
fibD1 = 7.3; pFibD1 = 20;
fibD2 = 8.7; pFibD2 = 10;
fibD3 = 10.0; pFibD3 = 5;
fibD4 = 11.5; pFibD4 = 20;
fibD5 = 12.8; pFibD5 = 25;
fibD6 = 14.0; pFibD6 = 5;
fibD7 = 15.0; pFibD7 = 3;
fibD8 = 16.0; pFibD8 = 2;
}

//*********************RESET_DIST*****************************
//DESCRIPTION: resets default fiber diameters and distributions
//*********passed parameters: none
//*********returns: void
//**********************************************************
void reset_dist()
{
fibD0 = 0; pFibD0 = 0;
fibD1 = 0; pFibD1 = 0;
fibD2 = 0; pFibD2 = 0;
fibD3 = 0; pFibD3 = 0;
fibD4 = 0; pFibD4 = 0;
fibD5 = 0; pFibD5 = 0;
fibD6 = 0; pFibD6 = 0;
fibD7 = 0; pFibD7 = 0;
fibD8 = 0; pFibD8 = 0;
}

//*********************CREATE_MATRIX**************************
//DESCRIPTION:
//*********passed parameters: none
//*********returns: void
//**********************************************************

void create_matrix()
{
```

```
int i=0,j=0, k=0,l=0;
double stimfreq, tstop, halfcycle, cycle, timestep, restcycle, pulsewidth, stimnum,
temp, temp2;
char word[10];
char word2[10];
char * pEnd;

cout << "Enter stimulation frequency (Hz): ";
      cin>>stimfreq;
      // 600 Hz
      cout << "Enter stimulation pulse width (in msec): ";
      cin>>pulsewidth;
      // 0.5 ms
      cout << "Enter simulation stop time (in msec): ";
      cin >>tstop;
      // 10 ms

      cycle = 1000/(stimfreq);
      //600hz cycle =1.6666
      halfcycle = cycle/2;
      //600hz halfcycle =0.833
      timestep = tstop/halfcycle;
      //600hz timestep = 12
      restcycle = (1000/stimfreq)-pulsewidth;
      //600hz cycle =1.1666

      std::ifstream is;
      std::ofstream os;

      is.open("potentials.txt"); // Opens the input file
      os.open("potentialsmatrix.txt"); //Opens the output file

if (! is.is_open()){ cout << "Error opening is file"; exit (1); }

if (! os.is_open()){ cout << "Error opening os file"; exit (1); }

//read the first value (number of stimulation)
is >> word;
stimnum = strtod (word,&pEnd);
temp=restcycle;
temp2=cycle;
os<<word<<" "<< timestep <<endl;
streampos here = is.tellg();
is.close();
```

```
for (i; i<timestep/2; i++)
{
   os<<temp<<" ";
   is.open("potentials.txt");
   if (! is.is_open()){ cout << "Error opening is file"; exit (1); }
   is>>word2;
   while ( j!=(stimnum) )
      { // keep reading until end-of-file
         is >> word2;
         os<<word2<<" ";
         j++;
      }
j=0;
is.close();
is.clear();
os<<endl;
os<<temp2;
  while (k!=(stimnum))
   {
      os<<" "<<0;
      k++;
   }

   k=0;
   os<<endl;

   temp= temp2+restcycle;
   temp2= temp + pulsewidth;
}

//close file

os.close();

}
```

```
//************************EXIT*****************************
//
//
//
//************************************************************

int exit()
{
	char ans;
	int flag = -1;

	for(int i=0; i<6; i++)
	{
		if(step[i]==false)
			flag = i;
	}
	if(flag!=-1)
	{
		colorText(3);
		cout<<"WARNING!!!"<<endl;
		colorText(4);
		cout<<"Are you sure you want to exit without ";
		switch(flag)
		{
		case 0:
			cout<<"creating fascicles? (y/n) ";
			break;
		case 1:
			cout<<"exporting locus file? (y/n) ";
			break;
		case 2:
			cout<<"creating finite element mesh? (y/n) ";
			break;
		case 3:
			cout<<"creating electrode locations? (y/n) ";
			break;
		case 4:
			cout<<"creating measurement locations? (y/n) ";
			break;
		case 5:
			cout<<"creating potentials matrix? (y/n) ";
			break;
		}
		cin>>ans;
```

```
            if(ans=='n')
                return 0;
        }
        colorText(5);
        cout<<"\nTHANK YOU FOR USING PERIPHERAL NERVE BUILDER!!!\n\n"
            <<"Written by Sleiman Bou-Sleiman && Germain Aoun"<<endl;
        cout<<"\nECE Department - American University of Beirut - Beirut, Lebanon
\n\t\twww.aub.edu.lb\n";
        cout<<"Modified and improved by Simon Fouquette - Montreal's University ";
        cout<<"Dominique Pâquet-Ferron, M.Sc.A. "<<endl;
        cout<<"Polystim research group - Ecole Polytechnique Montreal - Montreal,
Canada\n\t\twww.polystim.polymtl.ca\n"<<endl;
        return 11;
}
//************************MAIN*****************************
//DESCRIPTION: Main routine for the program. User selects one
//          of the following options (add a fascicle, delete/
//          undelete fascicle, view fascicle info, create
//          locus file, create stat file, create FE mesh, exit).
//*********passed parameters: none
//*********returns: void
//*************************************************************
int main(void)
{
        for (int i=0; i<6; i++){
            step[i] = false;}

        srand ( time(NULL) );
        int choice=-1;
        char ans = 'p';
        while(choice!=11)
        {
            system("cls");
        colorText(1);
        cout<<"[[[P E R I P H E R A L   N E R V E   B U I L D E R]]]\nAXON
POSITION GENERATOR AND FINITE ELEMENT MESH BUILDER"<<endl;
        cout<<"**********************MAIN
MENU**********************"<<endl;
        colorText(2);
        cout<<"1.  Add  fascicle\t\t[fascicle  number:  "<<fasc_num<<"]\n2.
Delete/Undelete fascicle\n"
            <<"3. View fascicle info\n4. Create locus file\n5. Create statistics file"
            <<"\n6. Create mesh\n7. Create electrode\n8. Add tri polar reading
electrode\n9. Add measurement sites"
```

```
<<"\n10. Create potentials matrix\n11. Exit"<<endl;
   colorText(0);
   cout<<"Choice: ";
   cin>>choice;
         switch(choice)
         {
         case 1:
               while (ans!='n'||ans!='N'||ans!='d'||ans!='D'||ans!='l'||ans!='L')
               {

               colorText(3);
               cout<<"WARNING!!!!"<<endl;
               colorText(4);
               cout<<"DO NOT CREATE NEW DISTRIBUTION UNLESS
NEURON CODE IS MODIFIED...\nCURRENT CODE WORKS"
                     <<" ONLY WITH DEFAULT VALUES DESCRIBED BY
MCINTYRE ET AL."<<endl;

               colorText(5);
               cout<<"\nSTEP I: DEFINE DISTRIBUTION FOR NEW
FASCICLE"<<endl;

               colorText(0);
               cout<<"::::use default (d)/last used (l)/or new (n) axon diameter
and distribution? ";
               cin>>ans;

               if(ans=='n'||ans=='N')
               {
                     reset_dist();
                     define_dist();
               }
               else if(ans=='d'||ans=='D'){ default_dist();}
               else if((ans=='l'||ans=='L') && fasc_num==0)
               {
                     colorText(4);
                     cout<<"NO PREVIOUS DATA...using Default...\n";
                     default_dist();
               }

               if(ans=='n'||ans=='N'||ans=='d'||ans=='D'||ans=='l'||ans=='L')
               {
               colorText(5);
               cout<<"STEP II: CREATE FASCICLE"<<endl;
```

```
                create_fasc();
                step[0] = true;
                break;//to exit while
                }
                }
                break;
        case 2:
                del_view_fasc(0);
                break;
        case 3:
                del_view_fasc(1);
                break;
        case 4:
                merge_files();
                colorText(5);
                cout<<"Locus file created!!.....press any key to continue"<<endl;
                getch();
                colorText(0);
                step[1] = true;
                break;
        case 5:
                create_stat();
                colorText(5);
                cout<<"Statistics file created!!.....press any key to
continue"<<endl;
                getch();
                colorText(0);
                break;
        case 6:
                create_mesh();
                step[2] = true;
                break;
        case 7:
                create_electrode();
                step[3] = true;
                break;
        case 8:
                create_reading_electrode();
                break;
        case 9:
                add_measurement_sites();
                step[4] = true;
                getch();
                colorText(0);
```

```
                    break;
                case 10:
                    create_matrix();
        step[5] = true;
        break;
                default:
                    choice=exit();
                    break;
                }
        }
        return 0;
}
```

Annexe II

CODE DU RÉSEAU DE SCIRun/BioPSE

```
# SCIRun Network v1.22.0


set name {Main}
set bbox {37 393 1393 2393}
set creationDate {Mon Oct 03 2005}
set creationTime {09:07:34}
set runDate {Mon Oct 03 2005}
set runTime {09:19:59}
set notes {}

# Create a SCIRun->DataIO->FieldReader Module
set m1 [addModuleAtPosition "SCIRun" "DataIO" "FieldReader" 465 393]
set Notes($m1) {reads tetvolmesh}
set Notes($m1-Position) {n}
set Notes($m1-Color) {white}

# Create a SCIRun->DataIO->MatrixReader Module
set m2 [addModuleAtPosition "SCIRun" "DataIO" "MatrixReader" 697 395]
set Notes($m2) {reads conductivity indices}
set Notes($m2-Position) {n}
set Notes($m2-Color) {white}

# Create a SCIRun->DataIO->MatrixReader Module
set m3 [addModuleAtPosition "SCIRun" "DataIO" "MatrixReader" 741 556]
set Notes($m3) {reads conductivity table}
set Notes($m3-Position) {n}
set Notes($m3-Color) {white}

# Create a SCIRun->FieldsData->ManageFieldData Module
set m4 [addModuleAtPosition "SCIRun" "FieldsData" "ManageFieldData" 465
498]

# Create a BioPSE->Modeling->ModifyConductivities Module
set m5 [addModuleAtPosition "BioPSE" "Modeling" "ModifyConductivities"
467 678]

# Create a SCIRun->FieldsOther->SetProperty Module
set m6 [addModuleAtPosition "SCIRun" "FieldsOther" "SetProperty" 515
776]

# Create a BioPSE->Forward->SetupFEMatrix Module
set m7 [addModuleAtPosition "BioPSE" "Forward" "SetupFEMatrix" 645
1231]

# Create a SCIRun->Math->SolveMatrix Module
```

```
set m8 [addModuleAtPosition "SCIRun" "Math" "SolveMatrix" 817 1357]

# Create a SCIRun->FieldsOther->FieldMeasures Module
set m9 [addModuleAtPosition "SCIRun" "FieldsOther" "FieldMeasures" 864
766]
set Notes($m9) {cells/size}
set Notes($m9-Position) {n}
set Notes($m9-Color) {white}

# Create a SCIRun->FieldsData->ManageFieldData Module
set m10 [addModuleAtPosition "SCIRun" "FieldsData" "ManageFieldData"
822 872]

# Create a SCIRun->FieldsData->ChangeFieldDataAt Module
set m11 [addModuleAtPosition "SCIRun" "FieldsData" "ChangeFieldDataAt"
618 846]
set Notes($m11) {changed att. to node}
set Notes($m11-Position) {n}
set Notes($m11-Color) {white}

# Create a SCIRun->FieldsData->BuildInterpolant Module
set m12 [addModuleAtPosition "SCIRun" "FieldsData" "BuildInterpolant"
873 1026]

# Create a SCIRun->FieldsData->BuildInterpolant Module
set m13 [addModuleAtPosition "SCIRun" "FieldsData" "BuildInterpolant"
234 1058]

# Create a SCIRun->FieldsData->TransformData Module
set m14 [addModuleAtPosition "SCIRun" "FieldsData" "TransformData" 289
882]
set Notes($m14) {puts negative sign
for current mag}
set Notes($m14-Position) {def}
set Notes($m14-Color) {white}

# Create a SCIRun->DataIO->FieldReader Module
set m15 [addModuleAtPosition "SCIRun" "DataIO" "FieldReader" 1042 609]
set Notes($m15) {reads electrode location}
set Notes($m15-Position) {n}
set Notes($m15-Color) {white}

# Create a SCIRun->DataIO->MatrixReader Module
set m16 [addModuleAtPosition "SCIRun" "DataIO" "MatrixReader" 1228 609]
set Notes($m16) {reads electrode current}
set Notes($m16-Position) {n}
set Notes($m16-Color) {white}

# Create a SCIRun->FieldsData->ManageFieldData Module
set m17 [addModuleAtPosition "SCIRun" "FieldsData" "ManageFieldData"
1106 744]
```

```
# Create a BioPSE->Forward->ApplyFEMCurrentSource Module
set m18 [addModuleAtPosition "BioPSE" "Forward" "ApplyFEMCurrentSource"
1137 1244]
set Notes($m18) {apply cathode current}
set Notes($m18-Position) {def}
set Notes($m18-Color) {white}

# Create a BioPSE->Forward->ApplyFEMCurrentSource Module
set m19 [addModuleAtPosition "BioPSE" "Forward" "ApplyFEMCurrentSource"
426 1158]
set Notes($m19) {apply anode current}
set Notes($m19-Position) {def}
set Notes($m19-Color) {white}

# Create a SCIRun->DataIO->FieldReader Module
set m20 [addModuleAtPosition "SCIRun" "DataIO" "FieldReader" 204 607]

# Create a SCIRun->FieldsData->ManageFieldData Module
set m21 [addModuleAtPosition "SCIRun" "FieldsData" "ManageFieldData"
205 778]

# Create a SCIRun->FieldsData->ChangeFieldDataType Module
set m22 [addModuleAtPosition "SCIRun" "FieldsData"
"ChangeFieldDataType" 460 580]

# Create a SCIRun->FieldsData->ManageFieldData Module
set m23 [addModuleAtPosition "SCIRun" "FieldsData" "ManageFieldData"
148 1535]

# Create a SCIRun->Visualization->ShowField Module
set m24 [addModuleAtPosition "SCIRun" "Visualization" "ShowField" 415
1827]

# Create a SCIRun->Visualization->GenStandardColorMaps Module
set m25 [addModuleAtPosition "SCIRun" "Visualization"
"GenStandardColorMaps" 419 1512]

# Create a SCIRun->Visualization->RescaleColorMap Module
set m26 [addModuleAtPosition "SCIRun" "Visualization" "RescaleColorMap"
433 1666]

# Create a SCIRun->Render->Viewer Module
set m27 [addModuleAtPosition "SCIRun" "Render" "Viewer" 416 2349]

# Create a SCIRun->FieldsData->DirectInterpolate Module
set m28 [addModuleAtPosition "SCIRun" "FieldsData" "DirectInterpolate"
783 1728]

# Create a SCIRun->DataIO->FieldReader Module
set m29 [addModuleAtPosition "SCIRun" "DataIO" "FieldReader" 873 1461]
set Notes($m29) {read axon positions}
set Notes($m29-Position) {n}
set Notes($m29-Color) {white}
```

```
# Create a SCIRun->DataIO->MatrixReader Module
set m30 [addModuleAtPosition "SCIRun" "DataIO" "MatrixReader" 896 1524]

# Create a SCIRun->FieldsData->ManageFieldData Module
set m31 [addModuleAtPosition "SCIRun" "FieldsData" "ManageFieldData"
801 1624]

# Create a SCIRun->Visualization->ShowField Module
set m32 [addModuleAtPosition "SCIRun" "Visualization" "ShowField" 343
1383]

# Create a SCIRun->FieldsData->ManageFieldData Module
set m33 [addModuleAtPosition "SCIRun" "FieldsData" "ManageFieldData"
822 1840]

# Create a SCIRun->DataIO->MatrixWriter Module
set m34 [addModuleAtPosition "SCIRun" "DataIO" "MatrixWriter" 840 1978]
set Notes($m34) {Writes voltages at axon positions}
set Notes($m34-Position) {n}
set Notes($m34-Color) {white}

# Create a SCIRun->FieldsData->Gradient Module
set m35 [addModuleAtPosition "SCIRun" "FieldsData" "Gradient" 37 1698]

# Create a SCIRun->FieldsCreate->SampleField Module
set m36 [addModuleAtPosition "SCIRun" "FieldsCreate" "SampleField" 189
1704]

# Create a SCIRun->Visualization->StreamLines Module
set m37 [addModuleAtPosition "SCIRun" "Visualization" "StreamLines" 127
1803]

# Create a SCIRun->FieldsData->DirectInterpolate Module
set m38 [addModuleAtPosition "SCIRun" "FieldsData" "DirectInterpolate"
76 1894]

# Create a SCIRun->Visualization->GenStandardColorMaps Module
set m39 [addModuleAtPosition "SCIRun" "Visualization"
"GenStandardColorMaps" 231 2025]

# Create a SCIRun->Visualization->RescaleColorMap Module
set m40 [addModuleAtPosition "SCIRun" "Visualization" "RescaleColorMap"
157 2103]

# Create a SCIRun->Visualization->ShowField Module
set m41 [addModuleAtPosition "SCIRun" "Visualization" "ShowField" 108
2194]

# Create a SCIRun->Visualization->ShowField Module
set m42 [addModuleAtPosition "SCIRun" "Visualization" "ShowField" 573
2140]
```

```
# Create a SCIRun->Visualization->ShowField Module
set m43 [addModuleAtPosition "SCIRun" "Visualization" "ShowField" 745
2138]

# Create a SCIRun->Visualization->GenStandardColorMaps Module
set      m44      [addModuleAtPosition      "SCIRun"      "Visualization"
"GenStandardColorMaps" 438 1230]

# Create a SCIRun->Visualization->RescaleColorMap Module
set m45 [addModuleAtPosition "SCIRun" "Visualization" "RescaleColorMap"
409 1303]

# Create the Connections between Modules
set c1 [addConnection $m7 0 $m8 0]
set c2 [addConnection $m5 0 $m6 0]
set c3 [addConnection $m1 0 $m4 0]
set c4 [addConnection $m1 0 $m23 0]
set c5 [addConnection $m15 0 $m17 0]
set c6 [addConnection $m20 0 $m21 0]
set c7 [addConnection $m29 0 $m31 0]
set c8 [addConnection $m11 0 $m12 0]
set c9 [addConnection $m11 0 $m13 0]
set c10 [addConnection $m22 0 $m5 0]
set c11 [addConnection $m28 0 $m33 0]
set c12 [addConnection $m28 0 $m24 0]
set c13 [addConnection $m38 0 $m41 0]
set c14 [addConnection $m35 0 $m37 0]
set c15 [addConnection $m4 0 $m22 0]
set c16 [addConnection $m17 0 $m43 0]
set c17 [addConnection $m21 0 $m14 0]
set c18 [addConnection $m23 0 $m36 0]
set c19 [addConnection $m23 0 $m28 0]
set c20 [addConnection $m23 0 $m38 0]
set c21 [addConnection $m23 0 $m35 0]
set c22 [addConnection $m33 1 $m34 0]
set c23 [addConnection $m14 0 $m42 0]
set c24 [addConnection $m6 0 $m18 0]
set c25 [addConnection $m6 0 $m19 0]
set c26 [addConnection $m6 0 $m7 0]
set c27 [addConnection $m6 0 $m11 0]
set c28 [addConnection $m6 0 $m10 0]
set c29 [addConnection $m6 0 $m9 0]
set c30 [addConnection $m6 0 $m32 0]
set c31 [addConnection $m25 0 $m26 0]
set c32 [addConnection $m39 0 $m40 0]
set c33 [addConnection $m44 0 $m45 0]
set c34 [addConnection $m24 0 $m27 0]
set c35 [addConnection $m18 0 $m8 1]
set c36 [addConnection $m2 0 $m4 1]
set c37 [addConnection $m3 0 $m5 1]
set c38 [addConnection $m16 0 $m17 1]
set c39 [addConnection $m16 0 $m21 1]
set c40 [addConnection $m30 0 $m31 1]
set c41 [addConnection $m36 0 $m37 1]
```

```
set c42 [addConnection $m28 0 $m26 1]
set c43 [addConnection $m38 0 $m40 1]
set c44 [addConnection $m17 0 $m18 1]
set c45 [addConnection $m17 0 $m12 1]
set c46 [addConnection $m31 0 $m28 1]
set c47 [addConnection $m14 0 $m19 1]
set c48 [addConnection $m14 0 $m13 1]
set c49 [addConnection $m9 0 $m10 1]
set Disabled($c49) {1}
set c50 [addConnection $m6 0 $m45 1]
set c51 [addConnection $m8 0 $m23 1]
set c52 [addConnection $m26 0 $m24 1]
set c53 [addConnection $m40 0 $m41 1]
set c54 [addConnection $m45 0 $m32 1]
set c55 [addConnection $m32 0 $m27 1]
set c56 [addConnection $m37 0 $m38 1]
set c57 [addConnection $m12 0 $m18 2]
set c58 [addConnection $m13 0 $m19 2]
set c59 [addConnection $m41 0 $m27 2]
set c60 [addConnection $m19 0 $m18 3]
set c61 [addConnection $m42 0 $m27 3]
set c62 [addConnection $m43 0 $m27 4]

# Set GUI variables for the SCIRun->DataIO->FieldReader Module
set                                                    $m1-filename
{/share/peel/20/home/ferron/SCIRun/Projet/V2.4/mesh_tet.pts}
set $m1-filetype {TextTetVolField (*.pts,*.tet)}

# Set GUI variables for the SCIRun->DataIO->MatrixReader Module
set                                                    $m2-filename
{/share/peel/20/home/ferron/SCIRun/Projet/V2.4/mesh_cond_tet.con}
set $m2-filetype {TextColumnMatrix (*.*)}

# Set GUI variables for the SCIRun->DataIO->MatrixReader Module
set                                                    $m3-filename
{/share/peel/20/home/ferron/SCIRun/Projet/V2.4/mesh_cond_table.txt}
set $m3-filetype {TextDenseMatrix (*.*)}

# Set GUI variables for the BioPSE->Modeling->ModifyConductivities
Module
set $m5-num-entries {5}
set $m5-names-0 {matrix-row-1}
set $m5-sizes-0 {1.0}
set $m5-m00-0 {0.04}
set $m5-m01-0 {0.0}
set $m5-m02-0 {0.0}
set $m5-m10-0 {0.0}
set $m5-m11-0 {0.04}
set $m5-m12-0 {0.0}
set $m5-m20-0 {0.0}
set $m5-m21-0 {0.0}
set $m5-m22-0 {0.04}
set $m5-names-1 {matrix-row-2}
set $m5-sizes-1 {1.0}
```

```
set $m5-m00-1 {0.1}
set $m5-m01-1 {0.0}
set $m5-m02-1 {0.0}
set $m5-m10-1 {0.0}
set $m5-m11-1 {0.1}
set $m5-m12-1 {0.0}
set $m5-m20-1 {0.0}
set $m5-m21-1 {0.0}
set $m5-m22-1 {0.1}
set $m5-names-2 {matrix-row-3}
set $m5-sizes-2 {1.0}
set $m5-m00-2 {0.01}
set $m5-m01-2 {0.0}
set $m5-m02-2 {0.0}
set $m5-m10-2 {0.0}
set $m5-m11-2 {0.01}
set $m5-m12-2 {0.0}
set $m5-m20-2 {0.0}
set $m5-m21-2 {0.0}
set $m5-m22-2 {0.01}
set $m5-names-3 {matrix-row-4}
set $m5-sizes-3 {1.0}
set $m5-m00-3 {0.08}
set $m5-m01-3 {0.0}
set $m5-m02-3 {0.0}
set $m5-m10-3 {0.0}
set $m5-m11-3 {0.08}
set $m5-m12-3 {0.0}
set $m5-m20-3 {0.0}
set $m5-m21-3 {0.0}
set $m5-m22-3 {0.08}
set $m5-names-4 {matrix-row-5}
set $m5-sizes-4 {1.0}
set $m5-m00-4 {0.0001}
set $m5-m01-4 {0.0}
set $m5-m02-4 {0.0}
set $m5-m10-4 {0.0}
set $m5-m11-4 {0.0001}
set $m5-m12-4 {0.0}
set $m5-m20-4 {0.0}
set $m5-m21-4 {0.0}
set $m5-m22-4 {0.0001}

# Set GUI variables for the SCIRun->FieldsOther->SetProperty Module
set $m6-val {mm}

# Set GUI variables for the SCIRun->Math->SolveMatrix Module
set $m8-target_error {0.011}
set $m8-flops {169206912.0}
set $m8-floprate {188.456385682}
set $m8-memrefs {28891555392.0}
set $m8-memrate {3220.50719875}
set $m8-orig_error {1.0}
set $m8-current_error {0.0106503}
```

```
set $m8-iteration {471}
set $m8-maxiter {1036}
set $m8-status {}

# Set GUI variables for the SCIRun->FieldsOther->FieldMeasures Module
set $m9-simplexString {Cell}
set $m9-xFlag {0}
set $m9-yFlag {0}
set $m9-zFlag {0}
set $m9-sizeFlag {1}

# Set GUI variables for the SCIRun->FieldsData->TransformData Module
set $m14-function {result = -v;

}

# Set GUI variables for the SCIRun->DataIO->FieldReader Module
set                                                    $m15-filename
{/share/peel/20/home/ferron/SCIRun/Projet/V2.4/cuff_cath.pts}
set $m15-filetype {TextPointCloudField (*.pts)}

# Set GUI variables for the SCIRun->DataIO->MatrixReader Module
set                                                    $m16-filename
{/share/peel/20/home/ferron/SCIRun/Projet/V2.4/cuff_curr.txt}
set $m16-filetype {TextColumnMatrix (*.*)}

# Set GUI variables for the BioPSE->Forward->ApplyFEMCurrentSource
Module
set $m18-modeTCL {sources and sinks}

# Set GUI variables for the BioPSE->Forward->ApplyFEMCurrentSource
Module
set $m19-modeTCL {sources and sinks}

# Set GUI variables for the SCIRun->DataIO->FieldReader Module
set                                                    $m20-filename
{/share/peel/20/home/ferron/SCIRun/Projet/V2.4/cuff_anod.pts}
set $m20-filetype {TextPointCloudField (*.pts)}

# Set GUI variables for the SCIRun->FieldsData->ChangeFieldDataType
Module
set $m22-outputdatatype {int}

# Set GUI variables for the SCIRun->Visualization->ShowField Module
set $m24-nodes-on {0}
set $m24-edges-on {0}
set $m24-faces-on {0}
set $m24-scalars-on {1}
set $m24-has_scalar_data {1}
set $m24-text-use-default-color {0}
set $m24-text-show-data {0}
set $m24-node_display_type {Spheres}
set $m24-active_tab {Scalars}
set $m24-node_scale {0.0033}
```

```
set $m24-scalars_scale {0.151}
set $m24-interactive_mode {OnExecute}
set $m24-field-name {conductivity}

# Set GUI variables for the SCIRun->Visualization->RescaleColorMap
Module
set $m26-min {-0.0458111078909}
set $m26-max {0.0458112539235}

# Set GUI variables for the SCIRun->Render->Viewer Module
set $m27-ViewWindow_0-caxes {0}
set $m27-ViewWindow_0-raxes {1}
set $m27-ViewWindow_0-have_collab_vis {0}
set $m27-ViewWindow_0-view-eyep-x {2.1}
set $m27-ViewWindow_0-view-eyep-y {1.6}
set $m27-ViewWindow_0-view-eyep-z {11.5}
set $m27-ViewWindow_0-view-lookat-x {0.0}
set $m27-ViewWindow_0-view-lookat-y {0.0}
set $m27-ViewWindow_0-view-lookat-z {0.0}
set $m27-ViewWindow_0-view-up-x {0.0}
set $m27-ViewWindow_0-view-up-y {1.0}
set $m27-ViewWindow_0-view-up-z {0.0}
set $m27-ViewWindow_0-view-fov {20.0}
set $m27-ViewWindow_0-trackViewWindow0 {1}
set $m27-ViewWindow_0-global-light0 {1}
set $m27-ViewWindow_0-global-light1 {0}
set $m27-ViewWindow_0-global-light2 {0}
set $m27-ViewWindow_0-global-light3 {0}
set $m27-ViewWindow_0-lightColors {{1.0 1.0 1.0} {1.0 1.0 1.0} {1.0 1.0
1.0} {1.0 1.0 1.0}}
set $m27-ViewWindow_0-lightVectors {{ 0 0 1 } { 0 0 1 } { 0 0 1 } { 0 0
1 }}
set $m27-ViewWindow_0-bgcolor-r {0.0}
set $m27-ViewWindow_0-bgcolor-g {0.0}
set $m27-ViewWindow_0-bgcolor-b {0.0}
set $m27-ViewWindow_0-do_stereo {0}
set $m27-ViewWindow_0-ambient-scale {1.0}
set $m27-ViewWindow_0-diffuse-scale {1.0}
set $m27-ViewWindow_0-specular-scale {0.4}
set $m27-ViewWindow_0-emission-scale {1.0}
set $m27-ViewWindow_0-shininess-scale {1.0}
set $m27-ViewWindow_0-polygon-offset-factor {1.0}
set $m27-ViewWindow_0-polygon-offset-units {0.0}
set $m27-ViewWindow_0-point-size {1.0}
set $m27-ViewWindow_0-line-width {1.0}
set $m27-ViewWindow_0-sbase {0.4}
set $m27-ViewWindow_0-sr {1}
set $m27-ViewWindow_0-do_bawgl {0}
set $m27-ViewWindow_0-global-light {1}
set $m27-ViewWindow_0-global-fog {0}
set $m27-ViewWindow_0-global-debug {0}
set $m27-ViewWindow_0-global-clip {1}
set $m27-ViewWindow_0-global-cull {0}
set $m27-ViewWindow_0-global-dl {0}
```

```
set $m27-ViewWindow_0-global-type {Gouraud}
set $m27-ViewWindow_0-ortho-view {0}
set $m27-ViewWindow_0-currentvisual {0}
set "$m27-ViewWindow_0-conductivity Scalars (1)" {1}
set "$m27-ViewWindow_0-conductivity Scalars (2)" {1}
set "$m27-ViewWindow_0-Nodes (3)" {1}
set "$m27-ViewWindow_0-Edges (3)" {1}
set "$m27-ViewWindow_0-Faces (3)" {1}
set "$m27-ViewWindow_0-Nodes (4)" {1}
set "$m27-ViewWindow_0-Edges (4)" {1}
set "$m27-ViewWindow_0-Faces (4)" {1}
set "$m27-ViewWindow_0-Nodes (5)" {1}
set "$m27-ViewWindow_0-Edges (5)" {1}
set "$m27-ViewWindow_0-Faces (5)" {1}

# Set GUI variables for the SCIRun->DataIO->FieldReader Module
set                                                          $m29-filename
{/share/peel/20/home/ferron/SCIRun/Projet/V2.4/axonpos.pts}
set $m29-filetype {TextPointCloudField (*.pts)}

# Set GUI variables for the SCIRun->DataIO->MatrixReader Module
set                                                          $m30-filename
{/share/peel/20/home/ferron/SCIRun/Projet/V2.4/axondummy.cmat}
set $m30-filetype {TextColumnMatrix (*.*)}

# Set GUI variables for the SCIRun->Visualization->ShowField Module
set $m32-nodes-on {0}
set $m32-edges-on {0}
set $m32-faces-on {0}
set $m32-scalars-on {1}
set $m32-has_scalar_data {1}
set $m32-text-use-default-color {0}
set $m32-text-show-data {0}
set $m32-active_tab {Scalars}
set $m32-scalars_scale {0.3}
set $m32-field-name {conductivity}

# Set GUI variables for the SCIRun->DataIO->MatrixWriter Module
set                                                          $m34-filename
{/share/peel/20/home/ferron/SCIRun/Projet/V2.4/potentials.mat}
set $m34-exporttype {SCIRun Matrix Binary (*.mat)}

# Set GUI variables for the SCIRun->FieldsCreate->SampleField Module
set $m36-endpoints {1}
set $m36-endpoint0x {-0.701338006955}
set $m36-endpoint0y {-0.233779335652}
set $m36-endpoint0z {-1.42533450174}
set $m36-endpoint1x {0.701338006955}
set $m36-endpoint1y {0.350669003478}
set $m36-endpoint1z {-1.01622066435}
set $m36-widgetscale {0.0420802804173}
set $m36-maxseeds {50}
```

```
# Set GUI variables for the SCIRun->Visualization->StreamLines Module
set $m37-stepsize {0.8}
set $m37-tolerance {0.8}
set $m37-maxsteps {250}
set $m37-method {5}

# Set GUI variables for the SCIRun->Visualization->RescaleColorMap
Module
set $m40-min {-0.0458312368486}
set $m40-max {0.0458845616944}

# Set GUI variables for the SCIRun->Visualization->ShowField Module
set $m41-has_scalar_data {1}
set $m41-edge_display_type {Cylinders}
set $m41-active_tab {Edges}
set $m41-scalars_scale {0.3}

# Set GUI variables for the SCIRun->Visualization->ShowField Module
set $m42-has_scalar_data {1}
set $m42-scalars_scale {0.3}

# Set GUI variables for the SCIRun->Visualization->ShowField Module
set $m43-has_scalar_data {1}

# Set GUI variables for the SCIRun->Visualization->GenStandardColorMaps
Module
set $m44-width {376}
set $m44-height {40}
set $m44-mapType {8}
set $m44-minRes {13}

# Set GUI variables for the SCIRun->Visualization->RescaleColorMap
Module
set $m45-min {0.0}
set $m45-max {4.0}


::netedit scheduleok
```

Annexe III

CODE DES FICHIERS NEURON

AXNODE.hoc – Modèle de Nœud de Ranvier proposé par McIntyre et al. pour un axone myélinisé

```
INDEPENDENT {t FROM 0 TO 1 WITH 1 (ms)}

NEURON {
      SUFFIX axnode
      NONSPECIFIC_CURRENT ina
      NONSPECIFIC_CURRENT inap
      NONSPECIFIC_CURRENT ik
      NONSPECIFIC_CURRENT il
      RANGE gnapbar, gnabar, gkbar, gl, ena, ek, el
      RANGE mp_inf, m_inf, h_inf, s_inf
      RANGE tau_mp, tau_m, tau_h, tau_s
}

UNITS {
      (mA) = (milliamp)
      (mV) = (millivolt)
}

PARAMETER {

      gnapbar = 0.01     (mho/cm2)
      gnabar       = 3.0 (mho/cm2)
      gkbar    = 0.08     (mho/cm2)
      gl     = 0.007 (mho/cm2)
      ena      = 50.0   (mV)
      ek       = -90.0 (mV)
      el     = -90.0 (mV)
      celsius            (degC)
      dt               (ms)
      v                (mV)
      vtraub=-80
      ampA = 0.01
      ampB = 27
      ampC = 10.2
      bmpA = 0.00025
      bmpB = 34
      bmpC = 10
      amA = 1.86
      amB = 21.4
      amC = 10.3
      bmA = 0.086
```

```
        bmB = 25.7
        bmC = 9.16
        ahA = 0.062
        ahB = 114.0
        ahC = 11.0
        bhA = 2.3
        bhB = 31.8
        bhC = 13.4
        asA = 0.3
        asB = -27
        asC = -5
        bsA = 0.03
        bsB = 10
        bsC = -1
}

STATE {
        mp m h s
}

ASSIGNED {
        inap     (mA/cm2)
        ina    (mA/cm2)
        ik       (mA/cm2)
        il       (mA/cm2)
        mp_inf
        m_inf
        h_inf
        s_inf
        tau_mp
        tau_m
        tau_h
        tau_s
        q10_1
        q10_2
        q10_3
}

BREAKPOINT {
        SOLVE states METHOD cnexp
        inap = gnapbar * mp*mp*mp * (v - ena)
        ina = gnabar * m*m*m*h * (v - ena)
        ik   = gkbar * s * (v - ek)
        il   = gl * (v - el)
}
DERIVATIVE states {    : exact Hodgkin-Huxley equations
         evaluate_fct(v)
        mp'= (mp_inf - mp) / tau_mp
        m' = (m_inf - m) / tau_m
        h' = (h_inf - h) / tau_h
        s' = (s_inf - s) / tau_s
}
UNITSOFF
```

```
INITIAL {
:
:     Q10 adjustment
:

      q10_1 = 2.2 ^ ((celsius-20)/ 10 )
      q10_2 = 2.9 ^ ((celsius-20)/ 10 )
      q10_3 = 3.0 ^ ((celsius-36)/ 10 )

      evaluate_fct(v)
      mp = mp_inf
      m = m_inf
      h = h_inf
      s = s_inf
}

PROCEDURE evaluate_fct(v(mV)) { LOCAL a,b,v2

      a = q10_1*vtrap1(v)
      b = q10_1*vtrap2(v)
      tau_mp = 1 / (a + b)
      mp_inf = a / (a + b)

      a = q10_1*vtrap6(v)
      b = q10_1*vtrap7(v)
      tau_m = 1 / (a + b)
      m_inf = a / (a + b)

      a = q10_2*vtrap8(v)
      b = q10_2*bhA / (1 + Exp(-(v+bhB)/bhC))
      tau_h = 1 / (a + b)
      h_inf = a / (a + b)

      v2 = v - vtraub : convert to traub convention

      a = q10_3*asA / (Exp((v2+asB)/asC) + 1)
      b = q10_3*bsA / (Exp((v2+bsB)/bsC) + 1)
      tau_s = 1 / (a + b)
      s_inf = a / (a + b)
}

FUNCTION vtrap(x) {
      if (x < -50) {
            vtrap = 0
      }else{
            vtrap = bsA / (Exp((x+bsB)/bsC) + 1)
      }
}
FUNCTION vtrap1(x) {
      if (fabs((x+ampB)/ampC) < 1e-6) {
            vtrap1 = ampA*ampC
      }else{
            vtrap1 = (ampA*(x+ampB)) / (1 - Exp(-(x+ampB)/ampC))
```

```
        }}

FUNCTION vtrap2(x) {
        if (fabs((x+bmpB)/bmpC) < 1e-6) {
                vtrap2 = -bmpA*bmpC
        }else{
                vtrap2 = (bmpA*(-(x+bmpB))) / (1 - Exp((x+bmpB)/bmpC))
        }
}

FUNCTION vtrap6(x) {
        if (fabs((x+amB)/amC) < 1e-6) {
                vtrap6 = amA*amC
        }else{
                vtrap6 = (amA*(x+amB)) / (1 - Exp(-(x+amB)/amC))
        }
}

FUNCTION vtrap7(x) {
        if (fabs((x+bmB)/bmC) < 1e-6) {
                vtrap7 = -bmA*bmC
        }else{
                vtrap7 = (bmA*(-(x+bmB))) / (1 - Exp((x+bmB)/bmC))
        }
}

FUNCTION vtrap8(x) {
        if (fabs((x+ahB)/ahC) < 1e-6) {
                vtrap8 = -ahA*ahC
        }else{
                vtrap8 = (ahA*(-(x+ahB))) / (1 - Exp((x+ahB)/ahC))
        }
}

FUNCTION Exp(x) {
        if (x < -100) {
                Exp = 0
        }else{
                Exp = exp(x)
        }
}

UNITSON
```

MYE_AXON.hoc – Patron d'axone

```
//*********************************************************************
// MYELINATED AXON TEMPLATE...based on multicompartment double cable
// model of mammalian axon
//  Description: Defines a template for myelinated axons of the
// peripheral nervous system. This template can be called externally
// using different parameters
//
//                ____ _____ ____________________ _____ ____
//                 /     |      |                      |      |     \
//  =========      |      |                      |      |     ========
//    node    MYSA|FLUT  |        STIN          |FLUT  |MYSA   node
//  =========      |      |                      |      |     ========
//           \_____|______|______________________|______|____/
// node = Node of Ranvier
// MYSA = Myelin attachement segment
// FLUT = Paranodal main segments
// STIN = Internodal segments
//
// Written by: Sleiman Bou-Sleiman & Germain Aoun
// --Polystim research labs
//- Microelectronics Research Group - Ecole Polytechnique de Montreal,
//  Montreal, Canada
// --Electrical and Computer Engineering Department
//- Faculty of Engineering and Architecture - American University of
//  Beirut, Beirut, Lebanon
//
//*********************************************************************

begintemplate mye_axon

public node, MYSA, FLUT, STIN, middle, axonnodes, paranodes1,
paranodes2, axoninter
external v_init

create node[1], MYSA[1], FLUT[1], STIN[1]

proc model_globals() {  //defines global values for the model

      //morphological parameters//
fiberD=$1   //choose from 5.7, 7.3, 8.7, 10.0, 11.5, 12.8, 14.0, 15.0,
16.0
paralength1=3         //MYSA length (Berthold and Rydmark 1983)
nodelength=1.0

space_p1=0.002   //um
//MYSA periaxonal space width (Berthold & Rydmark 1983)
      space_p2=0.004   //um //FLUT periaxonal space width
      space_i=0.004    //um //STIN periaxonal space width

      //electrical parameters//
      rhoa=0.7e6 //Ohm-um//
      mycm=0.1 //uF/cm2/lamella membrane//
```

```
      mygm=0.001 //S/cm2/lamella membrane//
      }

//defines dependent variables based on fiber diameter
proc dependent_var() {local deltax, divisions

      if   (fiberD==5.7)   {g=0.605   axonD=3.4   nodeD=1.9   paraD1=1.9
paraD2=3.4 deltax=500 paralength2=35 nl=80}

      if   (fiberD==7.3)   {g=0.630   axonD=4.6   nodeD=2.4   paraD1=2.4
paraD2=4.6 deltax=750 paralength2=38 nl=100}

      if   (fiberD==8.7)   {g=0.661   axonD=5.8   nodeD=2.8   paraD1=2.8
paraD2=5.8 deltax=1000 paralength2=40 nl=110}

      if   (fiberD==10.0)   {g=0.690   axonD=6.9   nodeD=3.3   paraD1=3.3
paraD2=6.9 deltax=1150 paralength2=46 nl=120}

      if   (fiberD==11.5)   {g=0.700   axonD=8.1   nodeD=3.7   paraD1=3.7
paraD2=8.1 deltax=1250 paralength2=50 nl=130}

      if   (fiberD==12.8)   {g=0.719   axonD=9.2   nodeD=4.2   paraD1=4.2
paraD2=9.2 deltax=1350 paralength2=54 nl=135}

      if   (fiberD==14.0)   {g=0.739   axonD=10.4   nodeD=4.7   paraD1=4.7
paraD2=10.4 deltax=1400 paralength2=56 nl=140}

      if   (fiberD==15.0)   {g=0.767   axonD=11.5   nodeD=5.0   paraD1=5.0
paraD2=11.5 deltax=1450 paralength2=58 nl=145}

      if   (fiberD==16.0)   {g=0.791   axonD=12.7   nodeD=5.5   paraD1=5.5
paraD2=12.7 deltax=1500 paralength2=60 nl=150}

//computing extracellular resistivities: multiply rhoa which is the
//axoplasmic resistivity in ohm-m with a conversion factor of 0.01 (to
//make the result Mohm-cm) and then divide by the difference in area
//between the larger cylinder and the engulfed cylinder (which
comprises //the extracellular material)

      Rpn0=(rhoa*.01)/(PI*((((nodeD/2)+space_p1)^2)-((nodeD/2)^2)))

//extracellular resistivity for node
      Rpn1=(rhoa*.01)/(PI*((((paraD1/2)+space_p1)^2)-((paraD1/2)^2)))
      Rpn2=(rhoa*.01)/(PI*((((paraD2/2)+space_p2)^2)-((paraD2/2)^2)))
      Rpx=(rhoa*.01)/(PI*((((axonD/2)+space_i)^2)-((axonD/2)^2)))
      interlength=(deltax-nodelength-(2*paralength1)-(2*paralength2))/6

      //topological parameters//
      divisions = ($1/deltax)*1000
      axonnodes= int(divisions) //21
      paranodes1=2*(axonnodes-1)
      paranodes2=2*(axonnodes-1)
      axoninter=6*(axonnodes-1)
      }
```

```
proc init(){//$1 = fiber diameter, $2 = x, $3 = y, $4 = nerve length

      model_globals($1)
      dependent_var($4)

      create node[axonnodes], MYSA[paranodes1], FLUT[paranodes2],

STIN[axoninter]
      access node[0]

      currloc = 0 //current location in 3d reconstruction of axon
      currloc2 = 0 //location of end of compartment
      for i=0,axonnodes-1 { //create all nodes of ranvier
            node[i]{
                  nseg=1
                  diam=nodeD
                  L=nodelength
                  Ra=rhoa/10000
                  cm=2
                  insert axnode //insert ion channel mechanism

                  insert extracellular xraxial=Rpn0 xg=1e10 xc=0

                  //insert 3d location of compartment
                  pt3dclear()
                  pt3dadd($2,$3,currloc, diam)
                  currloc2=currloc+nodelength
                  pt3dadd($2,$3,currloc2, diam)
                  }
            //location of next node
            currloc=currloc+2*paralength1+2*paralength2+6*interlength
      }

      currloc = nodelength //starting location of MYSAs
      for i=0, paranodes1-1 {
            MYSA[i]{ //create all MYSAs
                  nseg=1
                  diam=fiberD
                  L=paralength1
                  Ra=rhoa*(1/(paraD1/fiberD)^2)/10000
                  cm=2*paraD1/fiberD
                  insert pas
                  g_pas=0.001*paraD1/fiberD
                  e_pas=v_init
                  insert   extracellular   xraxial=Rpn1   xg=mygm/(nl*2)
xc=mycm/(nl*2)

                  //insert 3d location of compartment
                  pt3dclear()
                  pt3dadd($2,$3,currloc, diam)
                  currloc2=currloc+paralength1
                  pt3dadd($2,$3,currloc2, diam)
                  }
```

```
            //calculates location of next MYSA segment
            if(i%2==0){
                  currloc=currloc+2*paralength2+6*interlength
            }else{
                  currloc=currloc+nodelength
            }
      }

      currloc=nodelength+paralength1
//starting location for FLUTs

      for i=0, paranodes2-1 {
            FLUT[i]{ //create all FLUTs
                  nseg=1
                  diam=fiberD
                  L=paralength2
                  Ra=rhoa*(1/(paraD2/fiberD)^2)/10000
                  cm=2*paraD2/fiberD
                  insert pas
                  g_pas=0.0001*paraD2/fiberD
                  e_pas=v_init
                  insert   extracellular   xraxial=Rpn2   xg=mygm/(nl*2)
xc=mycm/(nl*2)

                  //insert 3d location of compartment
                  pt3dclear()
                  pt3dadd($2,$3,currloc, diam)
                  currloc2=currloc+paralength2
                  pt3dadd($2,$3,currloc2, diam)
                  }
            //calculate location of next FLUT compartment
            if(i%2==0){
                  currloc=currloc+6*interlength
            }else{
                  currloc=currloc+nodelength+2*paralength1
            }
      }

      currloc=nodelength+paralength1+paralength2
//starting location for STINs
      for i=0, axoninter-1 {
            STIN[i]{ //create all STINs
                  nseg=1
                  diam=fiberD
                  L=interlength
                  Ra=rhoa*(1/(axonD/fiberD)^2)/10000
                  cm=2*axonD/fiberD
                  insert pas
                  g_pas=0.0001*axonD/fiberD
                  e_pas=v_init
                  insert   extracellular   xraxial=Rpx   xg=mygm/(nl*2)
                  xc=mycm/(nl*2)
```

```
//insert 3d location of compartment
                  pt3dclear()
                  pt3dadd($2,$3,currloc, diam)
                  currloc2=currloc+interlength
                  pt3dadd($2,$3,currloc2, diam)
                  }
            //calculate location of next STIN
            if((i+1)%6==0){

      currloc=currloc+2*paralength1+2*paralength2+nodelength
            }else{
                  currloc=currloc+interlength
            }

      }
      //connect all compartments together
      for i=0, axonnodes-2 {
            connect MYSA[2*i](0), node[i](1)
            connect FLUT[2*i](0), MYSA[2*i](1)
            connect STIN[6*i](0), FLUT[2*i](1)
            connect STIN[6*i+1](0), STIN[6*i](1)
            connect STIN[6*i+2](0), STIN[6*i+1](1)
            connect STIN[6*i+3](0), STIN[6*i+2](1)
            connect STIN[6*i+4](0), STIN[6*i+3](1)
            connect STIN[6*i+5](0), STIN[6*i+4](1)
            connect FLUT[2*i+1](0), STIN[6*i+5](1)
            connect MYSA[2*i+1](0), FLUT[2*i+1](1)
            connect node[i+1](0), MYSA[2*i+1](1)
            }
      middle = int(axonnodes/2)
}

endtemplate mye_axon
```

FASCICLE.hoc – Patron de fascicule

```
//*********************************************************************
// FASCICLE TEMPLATE*****************************************************
// Description: defines a fascicle by creating axons with different
// diameters and planar position.
// Calls the mye_axons template by supplying it with geometrical
// information (diameter, x and y positions, and passes it the desired
// nerve segmtn length)
//
// Input arguments: 1- number of axons (supplied by the driver program)
// Dependencies: depends on external objects and variable listed under
// 'external'. Most importantly needs handle to the axon coordinate
// file (locus.txt) and voltage file (potentials.txt)
//
// Written by: Sleiman Bou-Sleiman & Germain Aoun
//
// --Polystim research labs
// - Microelectronics Research Group - Ecole Polytechnique de Montreal,
//   Montreal, Canada
// --Electrical and Computer Engineering Department
// - Faculty of Engineering and Architecture - American University of
//   Beirut, Beirut, Lebanon
//
//
*********************************************************************

begintemplate fascicle

public mye_axons, nAxons
external axon_coord, axon_potentials, v_init, nervlen

objectvar mye_axons[1]

proc init(){
nAxons = $1
objectvar mye_axons[nAxons]
for i=0, nAxons-1{
      axo_diam = axon_coord.scanvar() //read diameter
      ex = axon_coord.scanvar() //read x position
      wy = axon_coord.scanvar() //read y position
      mye_axons[i] = new mye_axon(axo_diam, ex, wy, nervlen)
}

}
endtemplate fascicle
```

DRIVER.hoc – Fichier principal du modèle

```
load_file("nrngui.hoc") //load GUI interface

//*************Define global variables.....
celsius=37
v_init=-80 //mV//
nervlen=25 //mm//
threshold = 3 //mV//
//****************************************


load_file("mye_axon.hoc") //load the myelinated axon template

//load the input file that contains axon information

objref axon_coord, axon_potentials
axon_coord = new File()
axon_coord.ropen("locus.txt")

load_file("fascicle.hoc")
//always place this line after axon_coord has been opened

nFascs = axon_coord.scanvar() //get number of fascicles

objectvar fasc[nFascs]

      for i=0, nFascs-1{
//create fascicles with appropriate number of axons
            fasc[i] = new fascicle(axon_coord.scanvar())
      }

//**********Define stimulation parameters
dt=0.025 //ms//
tstop=175 //ms//
//*************************************

objref field, potentialsfile
potentialsfile = new File("potentialsmatrix.txt")
potentialsfile.ropen()
field = new Matrix()
stimnum = potentialsfile.scanvar() //get number of stimul
timestep = potentialsfile.scanvar() //get number of timestep

print "stimnum: ", stimnum
print "Timestep: ", timestep

field.scanf(potentialsfile,timestep,stimnum+1)
objref potentialsfile // free up space


objref vex[stimnum], tvec
```

```
tvec = field.getcol(0)

      for ii=0, stimnum-1{
            vex[ii] =  field.getcol(ii+1)
      }

objref field // free up space

//load measurementSiteMatrix file
objref z_matrix
z_matrix = new File()
z_matrix.ropen("increment.txt")

z_nb = z_matrix.scanvar()
z1 = z_matrix.scanvar()
z2 = z_matrix.scanvar()
increment =  (z2-z1)- (z2-z1)/2
z_pos = z1*1000
increment *=1000
inf_limit = z_pos - increment
sup_limit = z_pos + increment



objref noderef, mysaref, flutref, stinref
curr_pos = 0

for ii=0, z_nb-1{// loop over all z positions

for i=0, nFascs-1{//loops over all fascicles

for j=0, fasc[i].nAxons-1{//loops over all axons in fascicle

// ************* loop over all node in axon ***************
      for k=0, fasc[i].mye_axons[j].axonnodes-1{

            fasc[i].mye_axons[j].node[k] noderef = new SectionRef()

      for l=0, noderef.nchild-1 noderef.child[l]{
            for(x,0){
                  if(z3d(x) > inf_limit &&  z3d(x) <= sup_limit){

            for(x,0){vex[curr_pos].play(&e_extracellular(x), tvec )}
                  }
            }
            for(x,0){
                  if( z3d(x) > sup_limit ){k=99999999999999999}
            }
      }
      }



// ************* loop over all MYSA segments in axon ***************
```

```
for k=0, fasc[i].mye_axons[j].paranodes1-1{
      fasc[i].mye_axons[j].MYSA[k] mysaref = new SectionRef()

      for l=0, mysaref.nchild-1 mysaref.child[l]{
            for(x,0){
                  if(z3d(x) > inf_limit &&  z3d(x) <= sup_limit){

      for(x,0){vex[curr_pos].play(&e_extracellular(x), tvec )}
                  }
            }

            for(x,0){
                  if( z3d(x) > sup_limit ){k=99999999999999999}
            }
      }
}


// ************* loop over all FLUT segments in axon ****************
for k=0, fasc[i].mye_axons[j].paranodes2-1{

      fasc[i].mye_axons[j].FLUT[k] flutref = new SectionRef()

      for l=0, flutref.nchild-1 flutref.child[l]{
            for(x,0){
                  if(  z3d(x) > inf_limit &&  z3d(x) <= sup_limit){

      for(x,0){vex[curr_pos].play(&e_extracellular(x), tvec )}
                  }
            }

            for(x,0){
                  if( z3d(x) > sup_limit ){k=99999999999999999}
            }
      }
}


// ************* loop over all STIN segments in axon ****************
for k=0, fasc[i].mye_axons[j].axoninter -1{

      fasc[i].mye_axons[j].STIN[k] stinref = new SectionRef()

      for l=0, stinref.nchild-1 stinref.child[l]{
            for(x,0){
                  if(  z3d(x) > inf_limit &&  z3d(x) <= sup_limit){


      for(x,0){vex[curr_pos].play(&e_extracellular(x), tvec )}
                  }
            }

            for(x,0){
```

```
                    if( z3d(x) > sup_limit ){k=99999999999999999}
                }
        }
}
        curr_pos+=1
                }
        }
        z_pos+=2*increment
        inf_limit = z_pos-increment
        sup_limit = z_pos+increment
}

access fasc[0].mye_axons[0].node[0]
finitialize(v_init)
fcurrent()
```

INIT.hoc – Fichiers d'initialisation

```
xopen("$(NEURONHOME)/lib/hoc/noload.hoc")
xopen("driver.hoc")
xopen("start.ses")
run()
```

START.ses – Fichier d'initialisation de l'interface usager

```
objectvar save_window_, rvp_
objectvar scene_vector_[13]
objectvar ocbox_, ocbox_list_, scene_, scene_list_
{ocbox_list_ = new List()  scene_list_ = new List()}
{pwman_place(70,117,1)}

{
save_window_ = new Graph(0)
save_window_.size(0,100,-84.8917,45.1913)
scene_vector_[2] = save_window_
{save_window_.view(0,   -84.8917,   100,   130.083,   869,   123,   385.92,
208.96)}
graphList[0].append(save_window_)
save_window_.save_name("graphList[0].")
save_window_.addexpr("fasc[0].mye_axons[33].node[49].v(  0.5  )",  1,  1,
0.8, 0.9, 2)
save_window_.addexpr("fasc[0].mye_axons[11].node[32].v(  0.5  )",  2,  1,
0.8, 0.9, 2)
save_window_.addexpr("fasc[0].mye_axons[8].node[24].v(  0.5  )",  3,  1,
0.8, 0.9, 2)
save_window_.addexpr("fasc[0].mye_axons[18].node[20].v(  0.5  )",  4,  1,
0.8, 0.9, 2)
save_window_.addexpr("fasc[0].mye_axons[23].node[19].v(  0.5  )",  5,  1,
0.8, 0.9, 2)
save_window_.addexpr("fasc[0].mye_axons[32].node[17].v(  0.5  )",  6,  1,
0.8, 0.9, 2)
save_window_.addexpr("fasc[0].mye_axons[5].node[16].v(  0.5  )",  7,  1,
0.8, 0.9, 2)
save_window_.addexpr("fasc[0].mye_axons[13].node[16].v(  0.5  )",  8,  1,
0.8, 0.9, 2)
save_window_.addexpr("fasc[0].mye_axons[17].node[15].v(  0.5  )",  9,  1,
0.8, 0.9, 2)
}

{
save_window_ = new Graph(0)
save_window_.size(0,100,-84.8917,45.1913)
scene_vector_[3] = save_window_
{save_window_.view(0,   -84.8917,   100,   130.083,   869,   123,   385.92,
208.96)}
graphList[0].append(save_window_)
save_window_.save_name("graphList[0].")
```

```
save_window_.addexpr("fasc[1].mye_axons[39].node[49].v(  0.5  )",  1,  1,
0.8, 0.9, 2)
save_window_.addexpr("fasc[1].mye_axons[1].node[32].v(  0.5  )",  2,  1,
0.8, 0.9, 2)
save_window_.addexpr("fasc[1].mye_axons[15].node[24].v(  0.5  )",  3,  1,
0.8, 0.9, 2)
save_window_.addexpr("fasc[1].mye_axons[24].node[20].v(  0.5  )",  4,  1,
0.8, 0.9, 2)
save_window_.addexpr("fasc[1].mye_axons[45].node[19].v(  0.5  )",  5,  1,
0.8, 0.9, 2)
save_window_.addexpr("fasc[1].mye_axons[7].node[17].v(  0.5  )",  6,  1,
0.8, 0.9, 2)
save_window_.addexpr("fasc[1].mye_axons[32].node[16].v(  0.5  )",  7,  1,
0.8, 0.9, 2)
save_window_.addexpr("fasc[1].mye_axons[30].node[16].v(  0.5  )",  8,  1,
0.8, 0.9, 2)
save_window_.addexpr("fasc[1].mye_axons[10].node[15].v(  0.5  )",  9,  1,
0.8, 0.9, 2)
}

{
save_window_ = new Graph(0)
save_window_.size(0,100,-84.8917,45.1913)
scene_vector_[4] = save_window_
{save_window_.view(0,   -84.8917,   100,   130.083,   869,   123,   385.92,
208.96)}
graphList[0].append(save_window_)
save_window_.save_name("graphList[0].")
save_window_.addexpr("fasc[2].mye_axons[25].node[49].v(  0.5  )",  1,  1,
0.8, 0.9, 2)
save_window_.addexpr("fasc[2].mye_axons[3].node[32].v(  0.5  )",  2,  1,
0.8, 0.9, 2)
save_window_.addexpr("fasc[2].mye_axons[18].node[24].v(  0.5  )",  3,  1,
0.8, 0.9, 2)
save_window_.addexpr("fasc[2].mye_axons[11].node[20].v(  0.5  )",  4,  1,
0.8, 0.9, 2)
save_window_.addexpr("fasc[2].mye_axons[16].node[19].v(  0.5  )",  5,  1,
0.8, 0.9, 2)
save_window_.addexpr("fasc[2].mye_axons[7].node[17].v(  0.5  )",  6,  1,
0.8, 0.9, 2)
save_window_.addexpr("fasc[2].mye_axons[1].node[16].v(  0.5  )",  7,  1,
0.8, 0.9, 2)
}

{
save_window_ = new Graph(0)
save_window_.size(0,100,-84.8917,45.1913)
scene_vector_[5] = save_window_
{save_window_.view(0,   -84.8917,   100,   130.083,   869,   123,   385.92,
208.96)}
graphList[0].append(save_window_)
save_window_.save_name("graphList[0].")
save_window_.addexpr("fasc[3].mye_axons[22].node[49].v(  0.5  )",  1,  1,
0.8, 0.9, 2)
```

```
save_window_.addexpr("fasc[3].mye_axons[10].node[32].v(  0.5  )",  2,  1,
0.8, 0.9, 2)
save_window_.addexpr("fasc[3].mye_axons[5].node[24].v(  0.5  )",  3,  1,
0.8, 0.9, 2)
save_window_.addexpr("fasc[3].mye_axons[3].node[20].v(  0.5  )",  4,  1,
0.8, 0.9, 2)
save_window_.addexpr("fasc[3].mye_axons[18].node[19].v(  0.5  )",  5,  1,
0.8, 0.9, 2)
save_window_.addexpr("fasc[3].mye_axons[15].node[17].v(  0.5  )",  6,  1,
0.8, 0.9, 2)
save_window_.addexpr("fasc[3].mye_axons[9].node[16].v(  0.5  )",  7,  1,
0.8, 0.9, 2)
save_window_.addexpr("fasc[3].mye_axons[21].node[15].v(  0.5  )",  9,  1,
0.8, 0.9, 2)
}

{
save_window_ = new Graph(0)
save_window_.size(0,100,-84.8917,45.1913)
scene_vector_[6] = save_window_
{save_window_.view(0,  -84.8917,  100,  130.083,  869,  123,  385.92,
208.96)}
graphList[0].append(save_window_)
save_window_.save_name("graphList[0].")
save_window_.addexpr("fasc[4].mye_axons[8].node[49].v(  0.5  )",  1,  1,
0.8, 0.9, 2)
save_window_.addexpr("fasc[4].mye_axons[0].node[32].v(  0.5  )",  2,  1,
0.8, 0.9, 2)
save_window_.addexpr("fasc[4].mye_axons[7].node[20].v(  0.5  )",  4,  1,
0.8, 0.9, 2)
save_window_.addexpr("fasc[4].mye_axons[2].node[17].v(  0.5  )",  6,  1,
0.8, 0.9, 2)
save_window_.addexpr("fasc[4].mye_axons[4].node[15].v(  0.5  )",  9,  1,
0.8, 0.9, 2)
}

{
save_window_ = new Graph(0)
save_window_.size(0,100,-84.8917,45.1913)
scene_vector_[7] = save_window_
{save_window_.view(0,  -84.8917,  100,  130.083,  869,  123,  385.92,
208.96)}
graphList[0].append(save_window_)
save_window_.save_name("graphList[0].")
save_window_.addexpr("fasc[5].mye_axons[8].node[49].v(  0.5  )",  1,  1,
0.8, 0.9, 2)
save_window_.addexpr("fasc[5].mye_axons[17].node[32].v(  0.5  )",  2,  1,
0.8, 0.9, 2)
save_window_.addexpr("fasc[5].mye_axons[25].node[24].v(  0.5  )",  3,  1,
0.8, 0.9, 2)
save_window_.addexpr("fasc[5].mye_axons[5].node[20].v(  0.5  )",  4,  1,
0.8, 0.9, 2)
save_window_.addexpr("fasc[5].mye_axons[1].node[19].v(  0.5  )",  5,  1,
0.8, 0.9, 2)
```

```
save_window_.addexpr("fasc[5].mye_axons[16].node[17].v(  0.5  )",  6,  1,
0.8, 0.9, 2)
save_window_.addexpr("fasc[5].mye_axons[11].node[16].v(  0.5  )",  7,  1,
0.8, 0.9, 2)
save_window_.addexpr("fasc[5].mye_axons[7].node[16].v(  0.5  )",  8,  1,
0.8, 0.9, 2)
}

{
save_window_ = new Graph(0)
save_window_.size(0,100,-84.8917,45.1913)
scene_vector_[8] = save_window_
{save_window_.view(0,  -84.8917,  100,  130.083,  869,  123,  385.92,
208.96)}
graphList[0].append(save_window_)
save_window_.save_name("graphList[0].")
save_window_.addexpr("fasc[6].mye_axons[4].node[49].v(  0.5  )",  1,  1,
0.8, 0.9, 2)
save_window_.addexpr("fasc[6].mye_axons[11].node[32].v(  0.5  )",  2,  1,
0.8, 0.9, 2)
save_window_.addexpr("fasc[6].mye_axons[26].node[24].v(  0.5  )",  3,  1,
0.8, 0.9, 2)
save_window_.addexpr("fasc[6].mye_axons[16].node[20].v(  0.5  )",  4,  1,
0.8, 0.9, 2)
save_window_.addexpr("fasc[6].mye_axons[19].node[19].v(  0.5  )",  5,  1,
0.8, 0.9, 2)
save_window_.addexpr("fasc[6].mye_axons[8].node[17].v(  0.5  )",  6,  1,
0.8, 0.9, 2)
save_window_.addexpr("fasc[6].mye_axons[55].node[16].v(  0.5  )",  7,  1,
0.8, 0.9, 2)
save_window_.addexpr("fasc[6].mye_axons[37].node[16].v(  0.5  )",  8,  1,
0.8, 0.9, 2)
save_window_.addexpr("fasc[6].mye_axons[44].node[15].v(  0.5  )",  9,  1,
0.8, 0.9, 2)
}

{
save_window_ = new Graph(0)
save_window_.size(0,100,-84.8917,45.1913)
scene_vector_[9] = save_window_
{save_window_.view(0,  -84.8917,  100,  130.083,  869,  123,  385.92,
208.96)}
graphList[0].append(save_window_)
save_window_.save_name("graphList[0].")
save_window_.addexpr("fasc[7].mye_axons[18].node[49].v(  0.5  )",  1,  1,
0.8, 0.9, 2)
save_window_.addexpr("fasc[7].mye_axons[3].node[32].v(  0.5  )",  2,  1,
0.8, 0.9, 2)
save_window_.addexpr("fasc[7].mye_axons[14].node[24].v(  0.5  )",  3,  1,
0.8, 0.9, 2)
save_window_.addexpr("fasc[7].mye_axons[29].node[20].v(  0.5  )",  4,  1,
0.8, 0.9, 2)
save_window_.addexpr("fasc[7].mye_axons[24].node[19].v(  0.5  )",  5,  1,
0.8, 0.9, 2)
```

```
save_window_.addexpr("fasc[7].mye_axons[36].node[17].v(  0.5  )",  6,  1,
0.8, 0.9, 2)
save_window_.addexpr("fasc[7].mye_axons[8].node[16].v(  0.5  )",  7,  1,
0.8, 0.9, 2)
save_window_.addexpr("fasc[7].mye_axons[11].node[16].v(  0.5  )",  8,  1,
0.8, 0.9, 2)
save_window_.addexpr("fasc[7].mye_axons[6].node[15].v(  0.5  )",  9,  1,
0.8, 0.9, 2)
}

{
save_window_ = new PlotShape(0)
save_window_.size(-438.584,368.584,-130,250)
save_window_.variable("v")
scene_vector_[10] = save_window_
{save_window_.view(-438.584,  -130,  807.169,  380,  867,  406,  388.8,
183.04)}
fast_flush_list.append(save_window_)
save_window_.save_name("fast_flush_list.")
}

{
save_window_ = new Graph(0)
save_window_.size(0,7000,-80,40)
scene_vector_[11] = save_window_
{save_window_.view(0, -80, 20000, 120, 505, 408, 300.48, 200.32)}
flush_list.append(save_window_)
save_window_.save_name("flush_list.")
objectvar rvp_
rvp_ = new RangeVarPlot("v")
fasc[0].mye_axons[8].node[0] rvp_.begin(1)
fasc[0].mye_axons[8].node[17] rvp_.end(0)
rvp_.origin(391.5)
save_window_.addobject(rvp_, 2, 1, 0.8, 0.9)
}

{
save_window_ = new Graph(0)
save_window_.size(0,7000,-80,40)
scene_vector_[12] = save_window_
{save_window_.view(0, -80, 20000, 120, 505, 127, 300.48, 200.32)}
flush_list.append(save_window_)
save_window_.save_name("flush_list.")
objectvar rvp_
rvp_ = new RangeVarPlot("v")
fasc[0].mye_axons[4].STIN[0] rvp_.begin(1)
fasc[0].mye_axons[4].STIN[89] rvp_.end(0)
rvp_.origin(391.5)
save_window_.addobject(rvp_, 2, 1, 0.8, 0.9)
}
objectvar scene_vector_[1]
{doNotify()}
```

Annexe IV

GUIDE DE L'UTILISATEUR

Utilisation du « Peripheral Nerve Builder »

Le « Peripheral Nerve Builder » est un fichier exécutable sous le système d'opération MS Windows, il s'agit donc d'un « utilitaire » rédiger en C++ et compiler en exécutable; extension « .exe ». Afin de lancer l'utilitaire, il faut double-cliquer directement sur le fichier « PeripheralNerveBuilder.exe », ceci lancera une fenêtre de type MS-DOS à l'écran.

Interface usager du « Peripheral Nerve Builder »

11 choix sont proposés à l'utilisateur lui permettant de créer le modèle du nerf ainsi que tous les fichiers nécessaires pour réaliser une simulation. Voici ces choix :

1. Permet d'ajouter un ou des fascicules, on doit spécifier quelle distribution on veut utiliser, le rayon du fascicule que l'on veut ajouter ainsi que les coordonnées cartésiennes (en 2D) de son centre.

2. Permet d'effacer ou d'annuler l'effacement d'un fascicule

3. Permet de voir quelques informations relatives aux fascicules déjà créés tel le diamètre du fascicule, la position de son centre, le nombre total d'axones, une liste présentant le nombre d'axones de chaque diamètres ainsi que le pourcentage correspondant, l'aire totale du fascicule et le rapport de l'aire occupée par les axones sur l'aire totale

4. Permet de créer le fichier « locus.txt »

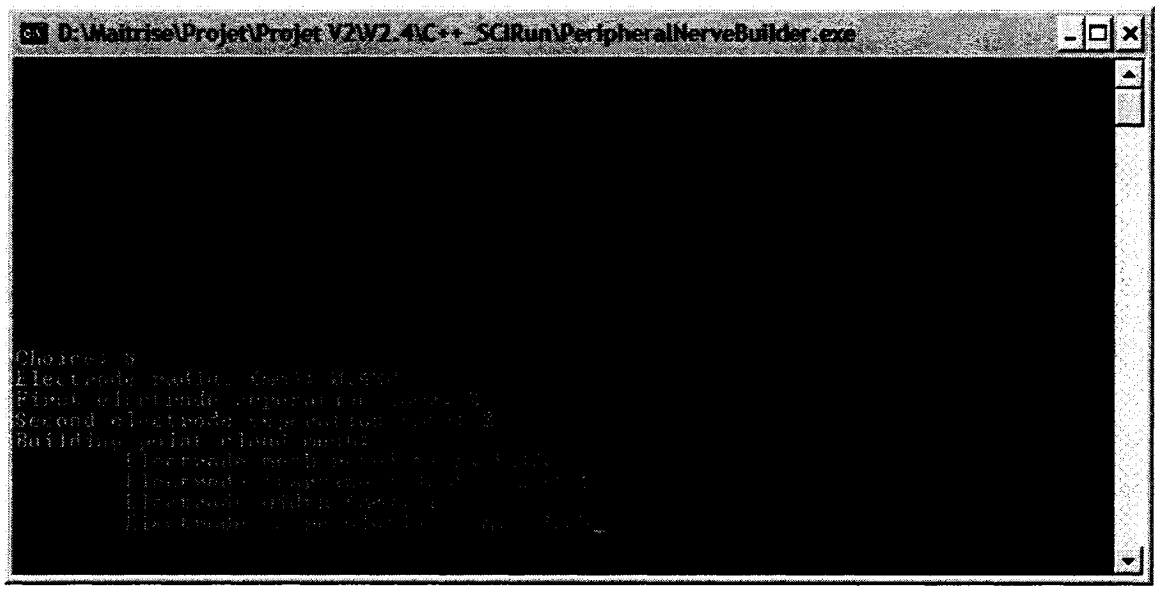
5. Permet de créer le fichier « statistics.txt »

6. Cette option permet de définir les paramètres du maillage; on doit spécifier la table contenant les conductivités, la forme du maillage (tétraédrique ou

hexaédrique), le rayon total du nerf, l'épaisseur de la couche de gras entourant le nerf, spécifier l'épaisseur pour le perineurium, la longueur totale du nerf ainsi que la résolution désiré en « x », en « y » et en « z ».

7. Permet de créer l'électrode; on doit spécifier le rayon et la séparation des contacts, l'amplitude du courant de stimulation, la résolution en « x » et en « y » du maillage composant les contacts, la position en « z » du premier contact, la largeur des contacts ainsi que la résolution en « z » du maillage de l'électrode.

8. Cette option permet de définir les différents paramètres pour une électrode tripolaire de lecture (cette option n'est pas implémentée dans SCIRun/BioPSE). On doit spécifier le rayon de l'électrode, l'espacement entre les contacts, la résolution en « x » et en « y » du maillage des contacts, la coordonnée en « z » du point d'insertion de l'électrode, la largeur des contacts ainsi que la résolution en « z » de l'électrode.



9. Cette option permet de définir la portion du nerf sur laquelle on veut extraire des données avec le réseau créer dans SCIRun/BioPSE. On doit donc spécifier un point de départ, un point d'arrêt ainsi que la résolution désirée.

10. Finalement, cette option est utilisée uniquement après avoir obtenu le fichier de sortie du réseau construit avec SCIRun/BioPSE « potentials.txt ». On spécifie les paramètres de stimulation pour la simulation finale avec Neuron. On choisit la fréquence du stimulus, la largeur des impulsions ainsi que la durée de simulation. Le fichier « potentialsmatrix.txt » est créé et servira d'entrée à la simulation avec Neuron.


D:\Maitrise\Projet\Projet V2\V2.4\C++_SCIRun\PeripheralNerveBuilder.exe
Choice: 10
Enter stimulation frequency (Hz): 300
Enter stimulation pulse width (in msec): 0.135
Enter simulation stop time (in msec): 250

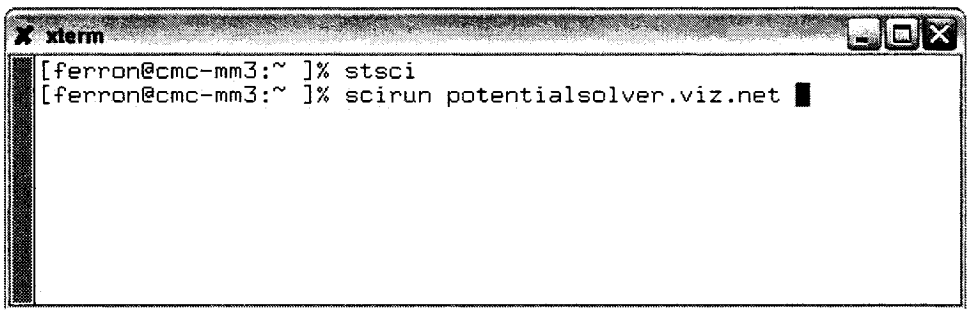
Utilisation du réseau construit avec SCIRun/BioPSE

Le réseau construit avec SCIRun/BioPSE permet d'extraire les valeurs de potentiels aux points d'intérêts spécifiés par le fichier créer avec l'option « 9 » du « Peripheral Nerve Builder » décrit à la section précédente.

Les fichiers nécessaires au fonctionnement du réseau sont, premièrement, le réseau lui-même, « potentialsolver.viz.net » et ceux générés par le « Peripheral Nerve Builder » :

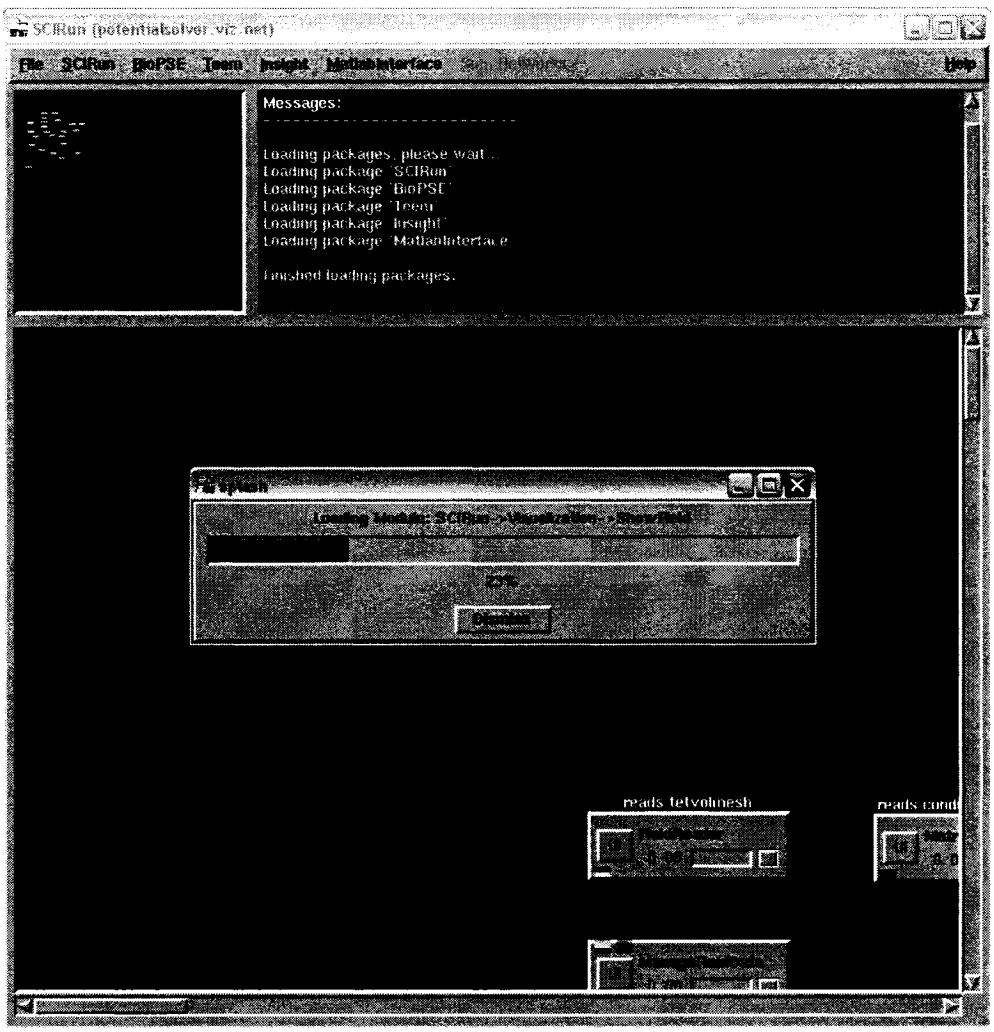
- Mesh_cond_table.txt : Contient la matrice des valeurs de conductivité des différents tissus.
- Mesh_cond_tet.con : Contient le nombre « N » d'éléments du maillage ainsi qu'une matrice de 1 X « N » contenant l'indice correspondant à la valeur de conductivité dans la table pour chaque élément.
- Mesh_tet.pts : Contient le nombre ainsi que les coordonnées « x, y et z » des nœuds du maillage.
- Mesh_tet.tet : Contient les indices des nœuds formant chaque tétraèdre.
- Axonpos.pts : Contient les coordonnées cartésiennes 3D des « M » points d'intérêts.
- Axondummy.cmat : Un fichier contenant une matrice nulle de 1 X « M ».
- Cuff_anod.pts : Contient le nombre de points constituant l'anode ainsi que leurs coordonnées cartésiennes 3D.
- Cuff_cath.pts : Idem mais pour la cathode.
- Cuff_curr.txt : Contient le nombre de points constituant un contact ainsi que la valeur du courant associée à chacun de ces points.

Le logiciel de résolutions des éléments finis SCIRun/BioPSE fonctionne sous « Linux », une fois tous les fichiers copier dans le répertoire de travail, on lance le logiciel en tapant à la ligne de commande :

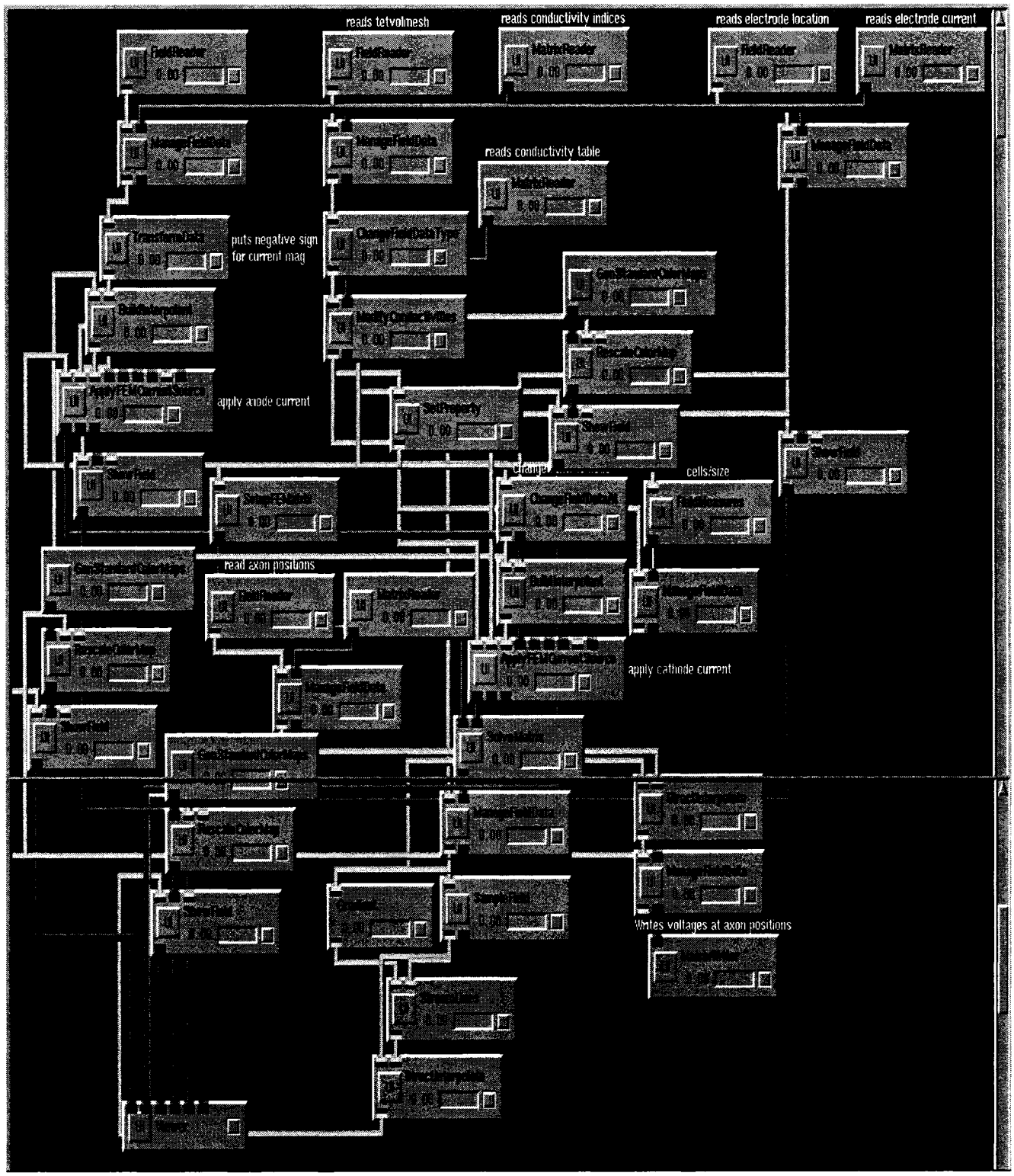


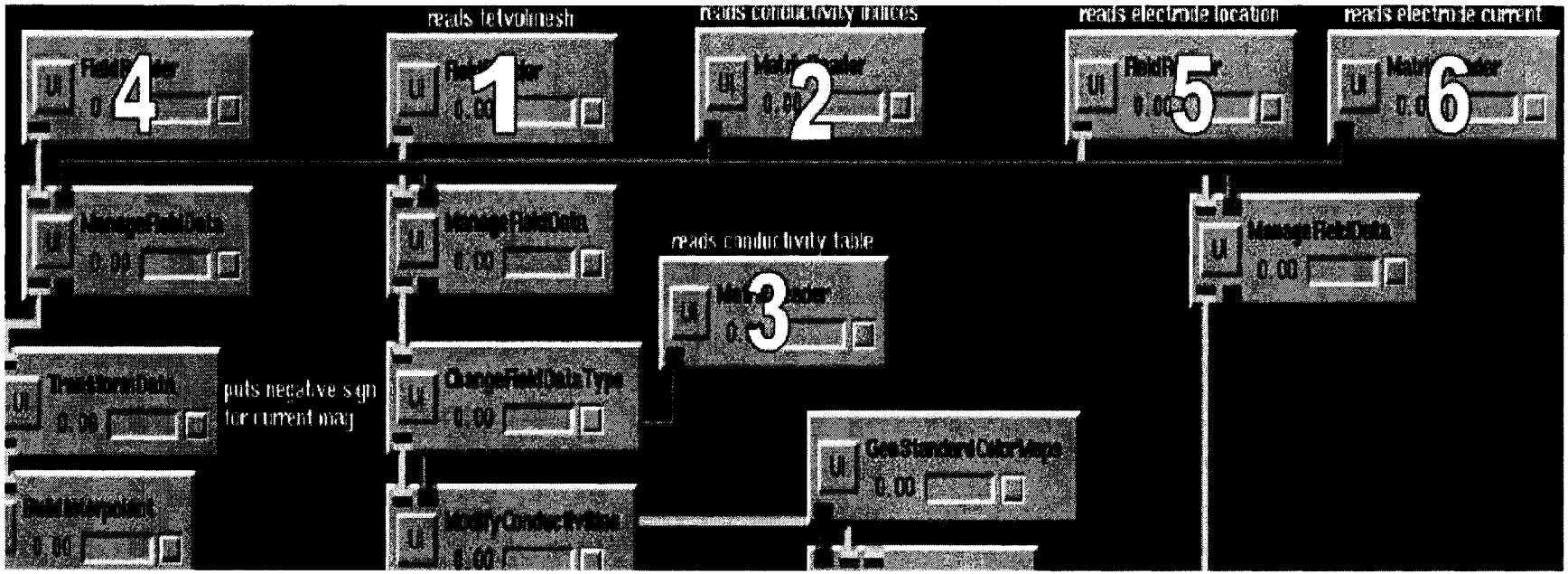
Démarrage de SCIRun/BioPSE

Initialisation du fichier contenant le réseau



Une fois le réseau initialisé, il faut associer les bons fichiers avec les bons blocs. Pour ce faire, on clique sur le bouton « UI » des blocs afin d'ouvrir une fenêtre permettant de naviguer et de sélectionner le fichier voulu. On répète l'opération pour chacun des blocs d'entrée.


reads tetvolmesh
reads conductivity indices
reads electrode location
reads electrode current
reads conductivity table
puts negative sign for current mag
apply anode current
cells/size
read axon positions
apply cathode current
Writes voltages at axon positions



	File	File Type
1	mesh_tet.pts	*TetVolField (*.pts, *tet)*
2	mesh_cond_tet.con	*ColumnMatrix (*.*)*
3	mesh_cond_table.txt	*DenseMatrix (*.*)*
4	cuff_anod.pts	*PointCloudField (*.pts)*
5	cuff_cath.pts	*PointCloudField (*.pts)*
6	cuff_curr.txt	*ColumnMatrix (*.*)*
7	axonposY.pts	*PointCloudField (*.pts)*
8	axondummy.cmat	*ColumnMatrix (*.*)*

Fig 1

Une fois tous les fichiers associés à leurs blocs respectifs, il ne reste plus qu'à exécuter le réseau afin d'obtenir et d'extraire les valeurs de potentiels; à partir du menu « File » sélectionner « Execute All ». Le fichier obtenu est « potentials.mat », il faut convertir ce fichier « .mat » en un fichier utilisable par Neuron à l'aide d'une fonction prédéfinie située dans les dossiers de SCIRun (SCIRun/bin/StandAlone/ convert). Directement à la ligne de commande Linux, on exécute la fonction de la façon suivante :

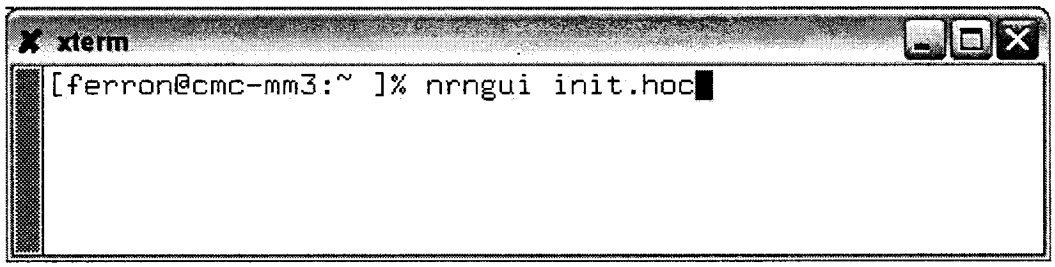
ColumnMatrixToText potentials.mat potentials.txt

Utilisation du logiciel Neuron et simulations

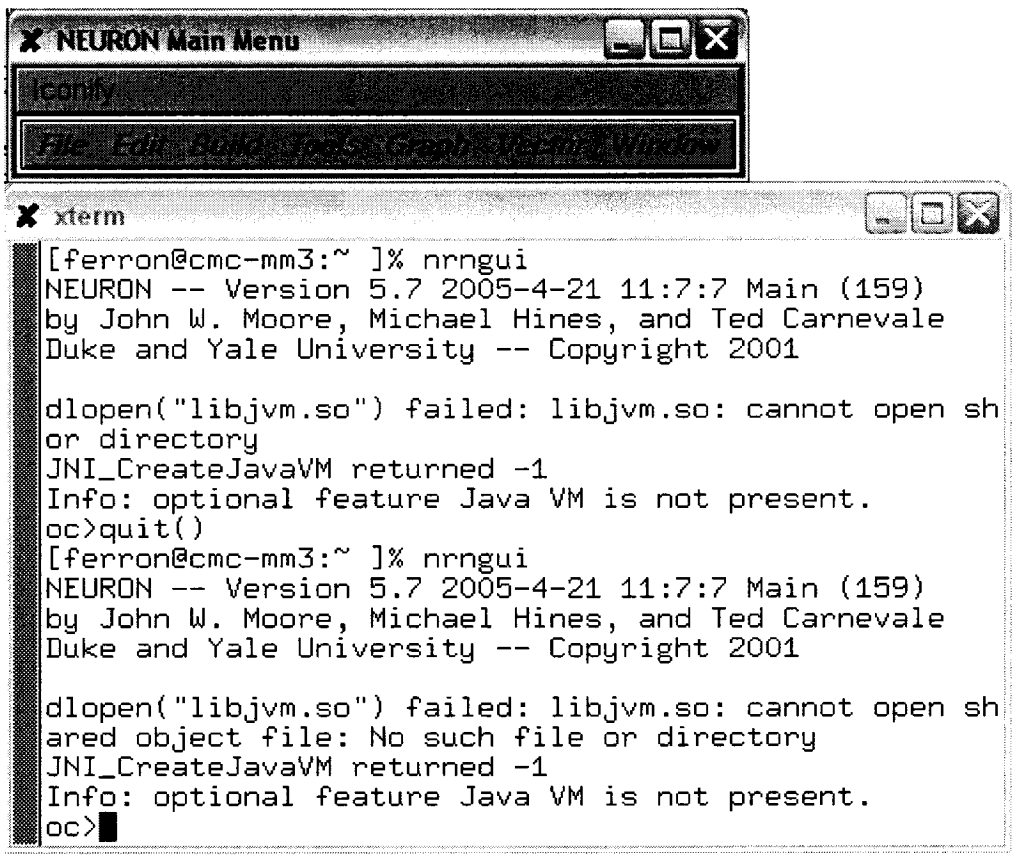
Le logiciel Neuron permet de simuler la propagation des PA au travers du nerf suite à l'application d'un courant de stimulation. Afin de démarrer une simulation à l'aide de Neuron, il faut tout d'abord que les fichiers suivants se retrouvent dans le répertoire de travail :

- AXNODE.hoc : Modèle des Nœuds de Ranvier.
- Mye_axon.hoc : Patron d'axone.
- Fascicle.hoc : Patron de fascicule.
- Driver.hoc : Fichier principal qui gère la création du modèle et l'ouverture de tous les autres fichiers nécessaires.
- Init.hoc : Fichier d'initialisation.
- Start.ses : Fichier d'initialisation de l'interface graphique.
- Increment.txt : Contient le nombre de points d'intérêt correspondant aux nombres de valeurs extraites ainsi que l'information permettant d'en déterminer leur position.
- Locus.txt : Contient le nombre de fascicule à générer, le nombre d'axones par fascicule et pour chacun de ces axones; son diamètre et s'est coordonnée en « x » et en « y ».
- Potentialsmatrix.txt : Fichier contenant les valeurs de potentiels extraites aux points d'intérêts.

Neuron peut fonctionner sous les systèmes d'exploitation suivants : Windows, Linux et Mac OS. La version utilisée dans ce projet est celle qui fonctionne sur Linux (voir raisons dans le chapitre 5 du présent mémoire). Ainsi pour démarrer Neuron il suffit de taper à la ligne de commande :

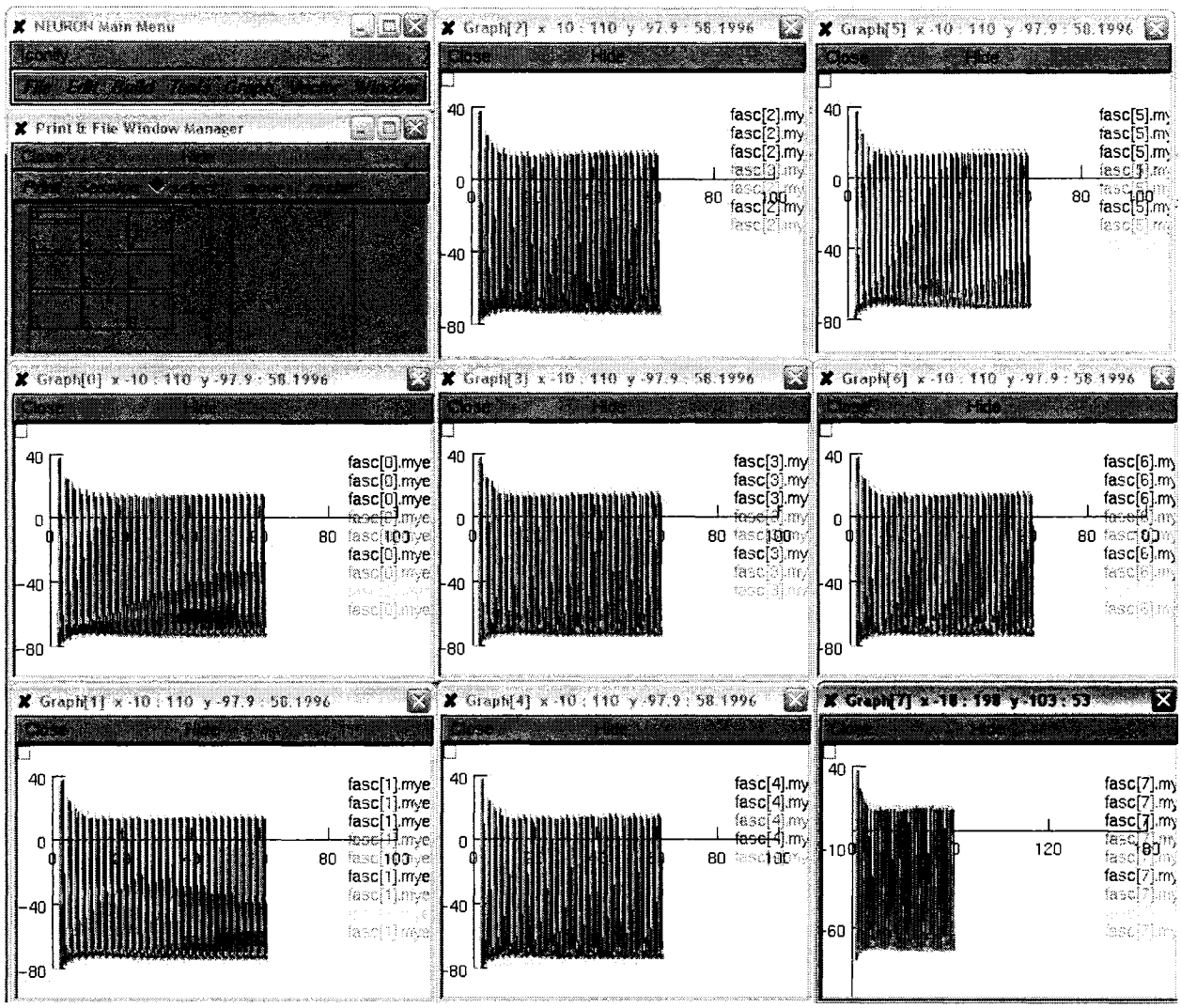


Démarrage de Neuron :



L'erreur que l'on aperçoit est « normal »; cela implique uniquement que l'option d'utilisation de « Java Virtual Machine » n'est pas disponible. Une fois le logiciel

démarré, on peut utiliser les différents menus pour démarrer une simulation, afficher des graphiques, etc. Cependant, une méthode beaucoup plus efficace consiste à utiliser le fichier « init.hoc », on peut modifier le fichier afin « d'automatiser » le démarrage d'une simulation. Ainsi, le démarrage de Neuron en utilisant le fichier « init.hoc » donne :



Tel que mentionné précédemment, afin de réaliser une simulation avec une fréquence ou une largeur d'impulsion différente, il faut retourner a l'option « 10 » du « Peripheral Nerve Builder ». Pour une simulation avec une amplitude différente, il faut recommencer du début et repasser au travers du réseau construit avec SCIRun/BioPSE.